applied biosystems™
by life technologies™

Applied Biosystems SOLiD™ System

BioScope™ Software for Scientists Guide

Data Analysis Methods and Interpretation

June 2010

For Research Use Use Only. Not intended for any animal or human therapeutic or diagnostic use.

APPLIED BIOSYSTEMS DISCLAIMS ALL WARRANTIES WITH RESPECT TO THIS DOCUMENT, EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO THOSE OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. TO THE FULLEST EXTENT ALLOWED BY LAW, IN NO EVENT SHALL APPLIED BIOSYSTEMS BE LIABLE, WHETHER IN CONTRACT, TORT, WARRANTY, OR UNDER ANY STATUTE OR ON ANY OTHER BASIS FOR SPECIAL, INCIDENTAL, INDIRECT, PUNITIVE, MULTIPLE OR CONSEQUENTIAL DAMAGES IN CONNECTION WITH OR ARISING FROM THIS DOCUMENT, INCLUDING BUT NOT LIMITED TO THE USE THEREOF, WHETHER OR NOT FORESEEABLE AND WHETHER OR NOT APPLIED BIOSYSTEMS IS ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

Notice to Purchaser: License Disclaimer

Purchase of this software product alone does not imply any license under any process, instrument or other apparatus, system, composition, reagent or kit rights under patent claims owned or otherwise controlled by Applied Biosystems, either expressly, or by estoppel.

TRADEMARKS:

Part Number 4448431 Rev. B
06/2010

Contents

1

Overview

This chapter covers:

SOLiD™ System overview

The Applied Biosystems SOLiD™ 4 System provides parallel sequencing of clonally amplified DNA fragments linked to magnetic beads. The sequencing methodology is based on sequential ligation with dye-labeled oligonucleotides. All fluorescently labeled oligonucleotide probes are present simultaneously, competing for incorporation. After each ligation, fluorescence is measured before another round of ligation takes place.

This ligation-based chemistry eliminates dephasing. The use of a two-base encoding mechanism, which interrogates each base twice for errors during sequencing, discriminates between true polymorphisms and system noise.

SOLiD™ component overview

The SOLiD™ system components include:

- SOLiD™ Analyzer and its ancillary equipment
- SOLiD™ Instrument Control Software (ICS)
- SOLiD™ Experiment Tracking System (SETS) software
- SOLiD™ BioScope™ Software

The SOLiD™ system includes the applications listed in Table 1 and in Figure 1 on page 17.

Table 1 SOLiD™ software applications

Software	Type	Function
ICS	Windows® operating system-based application	Instrument operation
SETS	Browser-based application	Reanalysis and reporting
BioScope™ Software	• Browser-based • Command line application	Secondary and tertiary analyses of primary analysis

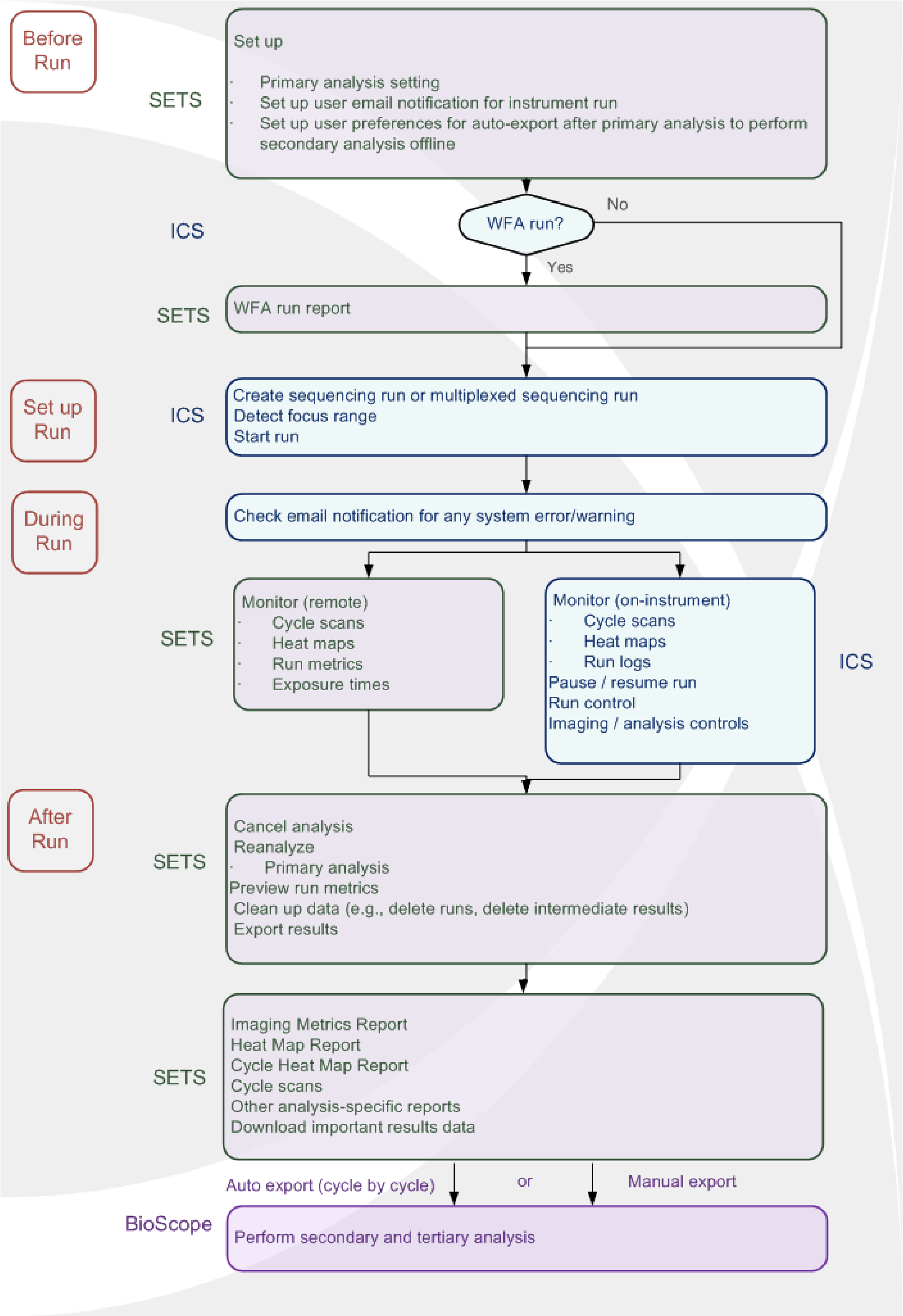

Before Run
SETS
Set up
Primary analysis setting
Set up user email notification for instrument run
Set up user preferences for auto-export after primary analysis to perform secondary analysis offline
ICS
WFA run?
No
Yes
SETS
WFA run report
Set up Run
ICS
Create sequencing run or multiplexed sequencing run
Detect focus range
Start run
During Run
Check email notification for any system error/warning
SETS
Monitor (remote)
Cycle scans
Heat maps
Run metrics
Exposure times
Monitor (on-instrument)
Cycle scans
Heat maps
Run logs
Pause / resume run
Run control
Imaging / analysis controls
ICS
After Run
SETS
Cancel analysis
Reanalyze
Primary analysis
Preview run metrics
Clean up data (e.g., delete runs, delete intermediate results)
Export results
SETS
Imaging Metrics Report
Heat Map Report
Cycle Heat Map Report
Cycle scans
Other analysis-specific reports
Download important results data
Auto export (cycle by cycle)
or
Manual export
BioScope
Perform secondary and tertiary analysis

Figure 1 SOLiD™ software workflow

BioScope™ Software overview

BioScope™ Software is used for secondary and tertiary data analysis. BioScope™ Software consists of a framework and a group of tools. The tools are executed and controlled through the framework via command-line input or the web interface.

For more information about secondary and tertiary analysis workflows, see "Secondary analysis workflow" on page 21 and "Tertiary analysis workflow" on page 23.

Note: A workflow comprises a series of analysis steps. Each analysis step is only dependent on the steps that precede it. For information about running a tool, see *Applied Biosystems SOLiD™ 4 System Software Integrated Workflow Quick Reference Guide (4448432)*. The document describes the relationship between the softwares comprising the SOLiD™ 4 platform and provides quick step procedures on operating each software to perform data analysis.

Installation options

Command line

This option installs all BioScope™ Software components that you access through a command line interface. Using the command line interface, you can build configuration files that can be used to assemble workflow tools.

Command line plus GUI

This option installs the command-line option plus a Tomcat web server that provides a Graphical User Interface (GUI). You can use GUI to modify or create configuration files that can be used to assemble workflow tools.

Full installation

This option installs the command-line, GUI, and the auto-export feature. Auto-export allows BioScope™ Software to automatically accept SOLiD™ data from SOLiD™ instruments on a cycle-by-cycle basis. Receiving data automatically removes the time-consuming step of copying the data at the end of each run.

As an option, you can install an examples directory. The directory provides BioScope™ Software sample applications, demonstration programs, configuration files, and more. You can install the examples directory before or after you install BioScope™ Software.

For more information

- See "Installing BioScope™ Software" on page 31
- see Appendix D, "Auto-Export" on page 325.
- see Appendix E, "Examples" on page 331

New features

New features in BioScope™ Software include:

- SOLiD™ Accuracy Enhancer Tool (SAET) - see "SAET" on page 19
- History tab - see "History tab" on page 19
- Barcode script - "Barcode script" on page 19
- *.bam format output - "*.bam format output" on page 19
- HD-300 performance - "HD-300 performance" on page 20
- Using SOLiDBioScope.com™ - see "Using SOLiDBioScope.com™" on page 20
- SNP detection post-error file changes - "SNP detection post-error file changes" on page 20
- Fusion/Splicing - "Fusion/Splicing" on page 20
- ChIP-Seq - "ChIP-Seq" on page 20
- Paired-end mapping - "Paired-end mapping" on page 20
- Export configuration - "Export configuration" on page 20

SAET

SAET is a spectral alignment error correction tool. When you apply the tool to raw data generated by the SOLiD™ platform, the color-calling error rate is reduced by up to five times, without the requirement of a reference genome. Using SAET can be an advantage because a decrease in the color calling error rate improves results in tools such as mapping, SNP calling, and *de novo* assembly results.

SAET is not recommended for use with whole genome resequencing of large genomes, where large is > 600 Mbases.

Use the SAET to preprocess raw reads before you begin mapping.

Note: SAET is only available through the BioScope™ Software command line.

For additional details about SAET, see Appendix B, "Use the SOLiD™ 4 Accuracy Enhancer Tool" on page 311.

History tab

The History tab in the BioScope™ Software GUI lets you download or view results files generated during a selected tool run. The History feature is not available via the command line.

Barcode script

The barcode script runs an analysis on a set of barcoded library read files in batch mode. Use the script to run simultaneous secondary or tertiary tests on barcoded libraries.

For additional details about the barcode script, see Appendix C, "Batch Analysis of Barcoded Library Data" on page 319.

*.bam format output

BioScope™ Software secondary analysis (mapping and pairing) produces a *.bam file as the main alignment format. Both mate-pair and paired-end analyses produce a *.bam file. A single file conversion is needed for fragment libraries.

For more information about the *.bam format output, see Appendix A, "File Format Descriptions" on page 295.

HD-300 performance

HD-300 is a SOLiD™ technology which increases throughput by allowing deposition densities of up to 300,000 beads per panel. BioScope™ Software can process larger files at increased throughput, which results in more reads while maintaining the established speed and increasing density.

Using SOLiDBioScope.com™

For information about this feature, see *Working with SOLiDBioScope.com™ Quick Reference Card (4452359)*. The document provides an online suite of software tools for Next Generation Sequencing (NGS) analysis. SOLiDBioScope.com™ leverages the scalable resources of cloud computing to perform compute-intensive NGS data processing.

SNP detection post-error file changes

BioScope™ Software provides the option of pre-generated Probe Error files. Providing pre-generated files allows you to save time by bypassing the regeneration of Probe Error files every time SNP detection is executed.

For additional details about SNP detection, see Chapter 11, "Run the Find SNPs Tool" on page 173.

Fusion/Splicing

A fusion junction is a section of transcribed RNA that maps to an exon from one gene followed by an exon from another gene. It can occur as the result of a translocation, deletion, or chromosomal inversion. It excludes exon-to-exon boundaries that arise from alternative splicing for a gene.

For more information about fusion/splicing or about fusion junctions, see Chapter 8, "Run the Find Splicing Fusion Tool" on page 109.

ChIP-Seq

BioScope™ Software includes the option to perform ChIP-Seq resequencing through the BioScope™ Software browser.

For more information about ChIP-Seq, see Chapter 16, "Run ChIP-Seq" on page 291.

Paired-end mapping

BioScope™ Software includes support for paired-end experiments in addition to mate-pair. In a normal paired-end experiment, the F5 and F3 tags match the genome on different strands facing toward each other, and these tags satisfy the distance constraint determined by insert size.

For more information about paired-end mapping, see "Run the Resequencing Pairing Tool" on page 149.

Export configuration

You now have the option to use either the BioScope™ Software web interface, or the command line, to create the configuration file required to perform experiments on barcoded libraries.

For more information about export configuration, see Appendix D, "Auto-Export" on page 325.

BioScope™ Software secondary and tertiary analysis workflow

Using BioScope™ Software allows you to perform secondary and tertiary analysis off-instrument. Figure 2 shows the flow of primary, secondary and tertiary analyses. Primary analysis is done on the instrument.

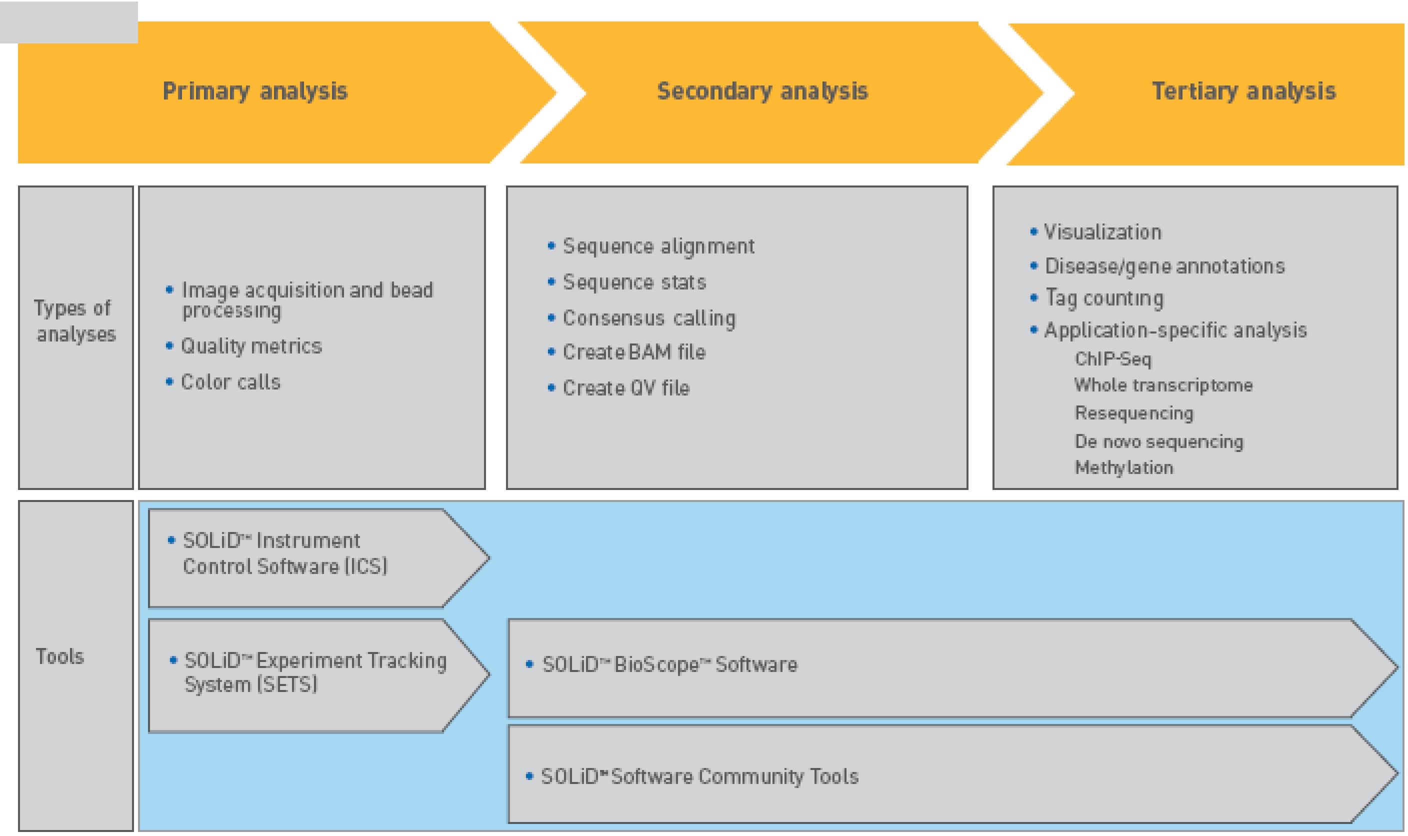


Figure 2 Primary, secondary, and tertiary analysis workflow

Secondary analysis workflow

Secondary analysis always starts with the mapping tool.

The input files required by the mapping tool are color-space fasta (*.csfasta) and quality value (*.qv).

The results of mapping and pairing in secondary analysis are used as input for tertiary analysis tools. You can only run the Inversion and Large Indel tools on the results from the pairing tool. The diBayes, CNV, and Small Indel tools can be executed on either mapping or pairing output. The Whole Transcriptome Analysis workflow is different from the resequencing and ChIP-Seq workflows because it uses fragment data to perform its own mapping with *.csfasta files as the primary file input.

BioScope™ Software secondary analysis features include:

- Mapping
- Mapping statistics
- Reporting

- Position errors
- Pairing

Secondary analysis consists of a set of serial steps of a modular workflow called an *analysis tool*. You can integrate a customized analysis tool into BioScope™ Software to automate secondary analysis and unify job management on the cluster.

The basic tools provided with the BioScope™ Software are resequencing and variation analysis tools.

During secondary analysis, BioScope™ Software performs the following steps:

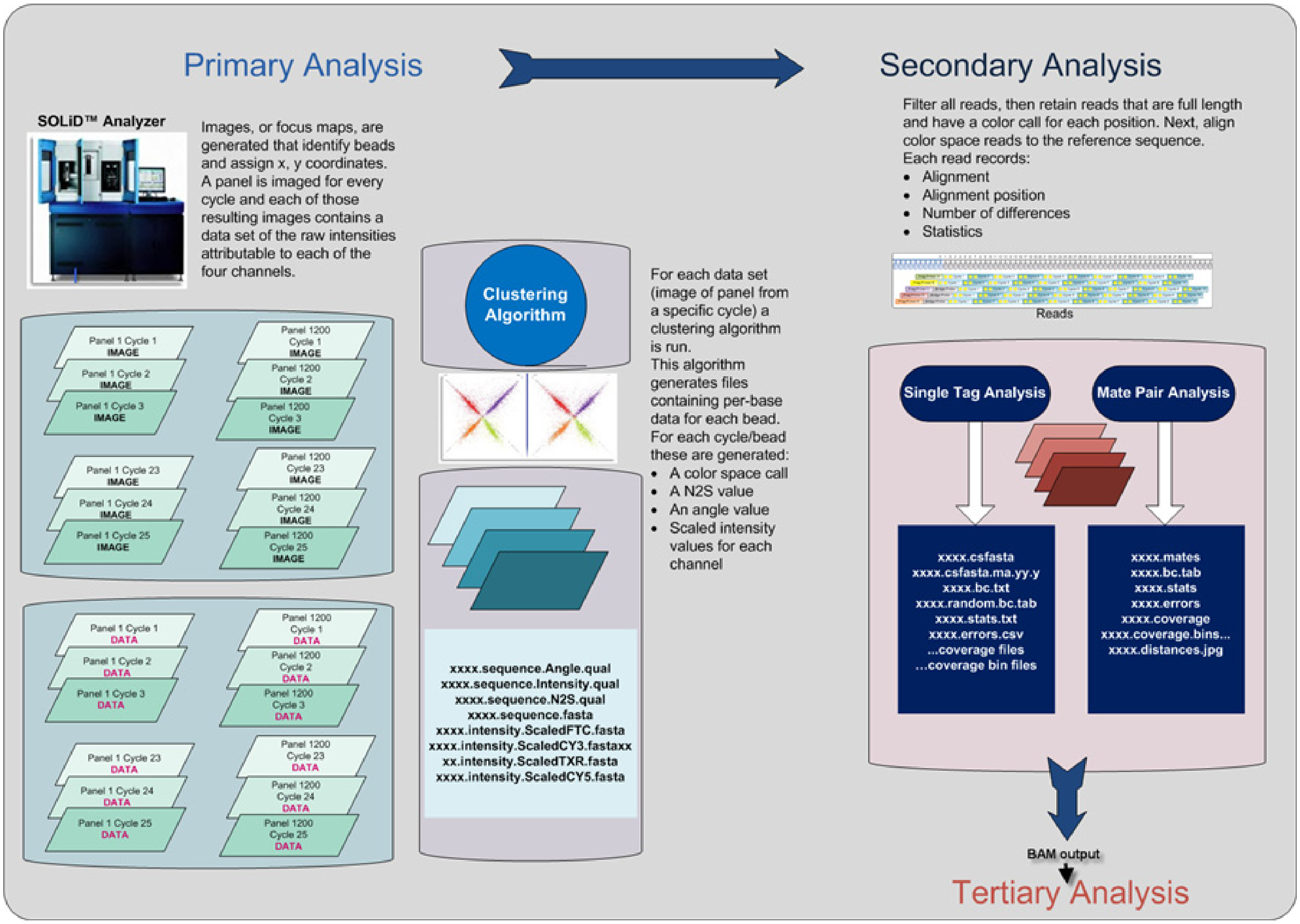
- Maps or aligns reads to a reference genome.
- Performs pairing for mate-pair runs and paired-end runs.
- Generates a *.bam file after completing mapping and pairing.

After the run is finished on the instrument, and the data from the run has been exported to BioScope™ Software, you can initialize analysis with BioScope™ Software through a command line or a Web browser by invoking parameter files.

Figure 3 on page 23 shows the workflow between primary and secondary analysis workflows.

For information about resequencing mapping, see Chapter 9, "Run the Resequencing Mapping Tool" on page 117. For information about resequencing pairing, see Chapter 10, "Run the Resequencing Pairing Tool" on page 149.

For information about Whole Transcriptome Analysis (WTA) mapping and pairing, see Chapter 5, "Run the Whole Transcriptome Data Mapping Tool" on page 83.



Images, or focus maps, are generated that identify beads and assign x, y coordinates. A panel is imaged for every cycle and each of those resulting images contains a data set of the raw intensities attributable to each of the four channels.

For each data set (image of panel from a specific cycle) a clustering algorithm is run. This algorithm generates files containing per-base data for each bead. For each cycle/bead these are generated:

- A color space call
- A N2S value
- An angle value
- Scaled intensity values for each channel

```
xxxx.sequence.Angle.qual
xxxx.sequence.Intensity.qual
xxxx.sequence.N2S.qual
xxxx.sequence.fasta
xxxx.intensity.ScaledFTC.fasta
xxxx.intensity.ScaledCY3.fastaxx
xx.intensity.ScaledTXR.fasta
xxxx.intensity.ScaledCY5.fasta
```

Filter all reads, then retain reads that are full length and have a color call for each position. Next, align color space reads to the reference sequence. Each read records:

- Alignment
- Alignment position
- Number of differences
- Statistics

```
xxxx.csfasta
xxxx.csfasta.ma.yy.y
xxxx.bc.txt
xxxx.random.bc.tab
xxxx.stats.txt
xxxx.errors.csv
...coverage files
...coverage bin files
```

```
xxxx.mates
xxxx.bc.tab
xxxx.stats
xxxx.errors
xxxx.coverage
xxxx.coverage.bins...
xxxx.distances.jpg
```

Figure 3 Primary and secondary analysis workflow

Tertiary analysis workflow

Tertiary analysis refers to analysis steps that generate biological interpretation. Examples include Find Inversions, Find SNPs, Count Known Exons.

Primary and secondary file types

See Table 2 for a list of files required for secondary and tertiary analysis.

Table 2 Primary and secondary file description

	File type	File name extension	File content
Primary analysis files	Raw reads file	*.csfasta	Color-space reads.
	QV quality value file	*.qual	Quality value for each color-space sequenced.
	Reads summary file	.stats	Statistics summarizing the number of reads collected in each panel on a slide.
	Scaled intensity value file (optional)	-intensity.scaled [CY3|CY3|CY5|FTC|TXR].fasta	Color-space reads.
Secondary analysis files	Mapping file	*.csfasta.ma	Sequence data mapped back to the reference sequence with quality values.
	*.bam	*.bam	Binary Alignment Map (BAM), a generic file format used to store large numbers of nucleotide sequence alignments.

Resequencing workflow overview

The resequencing tools in the following list are identified by their names in the BioScope™ Software web browser. Resequencing analysis includes the following tools:

- Map Data
- Find SNPs
- Find Human CNVs
- Find Inversions
- Find Large InDels
- Find Small InDels

See Table 3 on page 28 for a summary of the input and output files used by each resequencing tool.

Map data

The map data resequencing workflow uses *.csfasta files and *.qual files to create a *.bam file.

Find SNPs tool description

The diBayes package is the tool used to call Single Nucleotide Polymorphisms (SNPs) from mapped and processed SOLiD™ System color-space reads. The tool takes the color-space reads, the quality values, the reference sequence, and error information on each SOLiD™ slide as its input, and calls SNPs.

The tool creates three results files:

- A list of SNPs.
- A consensus *.fasta file with the same number of bases as the reference sequence (optional).
- A list of all covered positions (optional).

For information about using the SNP tool to run an experiment, see Chapter 11, "Run the Find SNPs Tool" on page 173.

Human CNV tool description

The human Copy Number Variation (CNV) tool detects Human CNV in SOLiD™ system data that originates from a single human sample. Slide(s) from this sample must be mapped to the hg18 reference to facilitate correct normalization.

IMPORTANT! The Human CNV tool in BioScope™ Software only includes normalization for humans, and so can only be used with human samples 'out-of-the-box'.

For information about using the Human CNV tool, see Chapter 12, "Run the Find Human CNVs Tool" on page 201.

Inversion tool description

An inversion is defined by its two breakpoints. The breakpoints are numbers of mate-pairs supporting the occurrence of the starting and ending inversion breakpoints that are counted for each base pair. The genomic ranges corresponding to local peaks of these counts, if above a score threshold, are called as candidate breakpoint ranges.

The tool is compatible with mate-pair data and calls inversions based on library size. The tool detects inversions using SOLiD™ mate files.

For information about using the Inversion tool to run an experiment, see "Run the Find Inversions Tool" on page 219.

Large Indel tool description

The Large Indel tool identifies deviations in clone insert size. These deviations indicate intrachromosomal structural variations compared to a reference genome. Insertions and deletions (indels) up to 100 kB are inferred by identifying positions in the genome in which the pairing distance between mapped mate-pairs is deviates significantly from what is expected at the given level of clone coverage.

A look-up table is created in which the amount that the clones must be deviated to achieve one standard deviation of significance is the standard error at each level of clone coverage. The table produces an asymptotic curve in which the minimum size of detectable indels at a given level of significance drops rapidly as the clone coverage increases. The look-up table is used to determine the significance of the deviation in average insert size at each position in the genome.

Regions of the genome that are significantly deviated are selected as candidate indels, and hierarchical clustering is used to segregate the clones into groups in which the difference in the sizes of all clones in a group is less than the specified range. Clusters with too few clones, as specified by the user, are removed and the candidates are assessed to determine if a homozygous or heterozygous population of deviated insert sizes remains. All clones deviated by ≥ 100 kB are discarded. Clones from various libraries with various insert sizes contribute to a single indel call by combining the probabilities associated with the clones from each library.

For information about using the Find Large Indel tool to run an experiment, see Chapter 14, "Run the Find Large InDels tool" on page 239.

Small Indel tool description

When an indel occurs in a sequence, and that sequence is measured using color-space, the color-space sequence has a gap the same size as the indel. The color-space sequence also leaves a signature that can indicate whether there is a measurement error within the gap. The Small Indel tool targets the processing of indel evidences found in the pairing step during secondary offline data analysis.

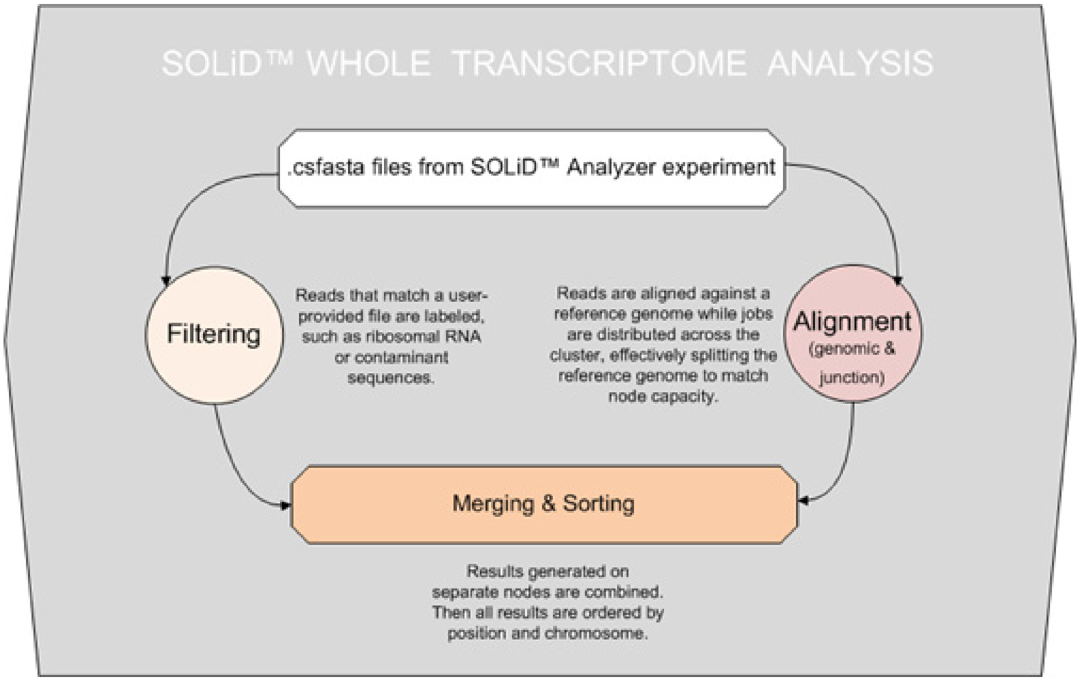
For information about using the Small Indel tool to run an experiment, see Chapter 15, "Run the Find Small InDels Tool" on page 263.

Whole Transcriptome Analysis workflow

High-throughput sequencing of the transcriptome using the SOLiD™ system enables genome-wide expression profiling with high sensitivity and a wider dynamic range than microarray technology. The Whole Transcriptome library preparation also preserves the strandedness of the RNA transcripts. Preserving the strandedness simplifies data analysis, allows determination of the directionality of transcription and gene orientation, and facilitates detection of opposing and overlapping transcripts (see Figure 4).

Workflow

The Whole Transcriptome Analysis (WTA) in BioScope™ Software aligns to a reference genome. Using the mapping results, WTA counts the number of tags aligned with exons, and can convert the *.bam file to a Wiggle File (*.wig) for display of coverage on the UCSC Genome Browser. WTA also supports experiments in fusion detection. For information about fusion detection, see Chapter 4, "Whole Transcriptome Pipeline Concepts" on page 47.


SOLiD™ WHOLE TRANSCRIPTOME ANALYSIS

.csfasta files from SOLiD™ Analyzer experiment

Filtering

Reads that match a user-provided file are labeled, such as ribosomal RNA or contaminant sequences.

Alignment (genomic & junction)

Reads are aligned against a reference genome while jobs are distributed across the cluster, effectively splitting the reference genome to match node capacity.

Merging & Sorting

Results generated on separate nodes are combined. Then all results are ordered by position and chromosome.

Figure 4 WTA workflow

Mapping transcriptome reads to a genome introduces complexities not found in traditional SOLiD™ genomic resequencing tools. Read-mapping for resequencing tools identifies correlated alignments between a read and the reference. Mapping from WTA could present gaps when a read crosses the exon-intron boundary.

Whole Transcriptome Analysis tools include:

- WT Map Data
- Create UCSC Wig File
- Count Known Exons
- Find Splicing Fusion

See Table 3 on page 28 for a summary of the input and output files used by each WTA tool.

WT Map Data

Four parallel read mappings occur in WTA:

- Mapping to filter sequences
- Mapping to splice junction
- Mapping to the genome reference
- Exon mapping and pairing

Mapping jobs are divided and distributed across the available cluster resources, then mapping results are merged and sorted. For details about mapping jobs, see "Run the Whole Transcriptome Data Mapping Tool" on page 83.

Create UCSC WIG file

This tool takes the *.bam file and converts it into *.wig files containing coverage data. Coverage is the number of reads covering a given genome stranded position.

For information about using the "Create UCSC WIG tool to run an experiment, see Chapter 7, "Run the Create UCSC WIG File Tool" on page 101.

Note: WIG files can be visualized in the UCSC Genome Browser.

Count known exons

The "Count Known Exons" tool generates tag counts for annotated regions. The required input files are a *.bam file of mapped reads, and a *.gtf file of predefined regions, such as exons.

For information about using the Count Known Exons tool to run an experiment, see "Run the Count Known Exons Tool" on page 93.

Fusion/splicing description

A fusion junction is a section of transcribed RNA that maps to an exon from one gene followed by an exon from another gene. It can occur as the result of a translocation, deletion, or chromosomal inversion. A fusion junction excludes exon-to-exon boundaries that arise from alternative splicing for a gene.

There are five models of alternative splicing:

- **Exon skipping or cassette exon:** In this model, an exon may be spliced out of the primary transcript or retained. This is the most common mode in mammalian pre-mRNAs.
- **Mutually exclusive exons:** One of two exons, but not both, is retained in mRNAs after splicing.
- **Alternative donor site:** An alternative 5' splice junction (donor site) is used, changing the 3' boundary of the upstream exon.

- **Alternative acceptor site:** An alternative 3' splice junction (acceptor site) is used, changing the 5' boundary of the downstream exon.
- **Intron retention:** A sequence may be spliced out as an intron or simply retained. This model is distinguished from exon skipping because the retained sequence is not flanked by introns. If the retained intron is in the coding region, the intron must encode amino acids in frame with the neighboring exons.

For information about using the Fusion/Splicing tool to run an experiment, see Chapter 8, "Run the Find Splicing Fusion Tool" on page 109.

ChIP-Seq workflow overview

This workflow includes incorporation of third-party tools and uses matching data as input.

For details, see Chapter 16, "Run ChIP-Seq" on page 291.

Note: ChIP-Seq and the resequencing tools use the same algorithm for mapping and pairing.

Input and output file formats

All tools use specific input files and produce specific output files. The input and output file requirements vary, depending on the type of analysis that you want to perform. Table 3 provides the names of the input and output file types.

Table 3 Input and output file format types for tools

Software or bioinformatics tool	Input file type(s)	Output file type
Mapping tool	*.csfasta, *.fasta, *qv	*.ma(local)
Pairing tool	*.ma(local) [,*.fasta], *.qual, *.csfasta	*.bam
MaToBam tool	—	Converts a *.ma file to a *.bam file.
Small indel	*.bam	*.gff.3
Frag indel	*.ma, *.ma(local)	*.pas
diBayes	*.bam	*.gff.3, *.csfasta, consensus_calls
CNV - singleSample	*.bam	*.gff.3
Large indel -singleSample	*.bam	*.gff.3
Large indel -pairedSample	—	—
Inversion	*.bam	*.gff.3, *.txt
Position error	*.bam	position error
WT mapping	*.csfasta, *fasta, filter reference fasta, WT *.gtf reference	*.bam
Counttag	*.bam	*.gtf
Sam2Wig	*.bam	.wig

Table 3 Input and output file format types for tools *(continued)*

Software or bioinformatics tool	Input file type(s)	Output file type
Key [ ] = optional + = 1 or more *.ma = classic match file *.ma(local) = match file with local alignment extensions *.gff.3 = public, viewer-oriented *.gff v 3 *.gff 0.2 = SOLiD™ *.gff version 2 *.gff 3.5 = SOLiD™ *.gff version for 3.5 release		

Visualizing *.bam files

The *.bam format is a generic format for storing large numbers of nucleotide sequence alignments. You can use third-party software visualization tools to view *.bam files in a Web browser.

See Appendix A, "File Format Descriptions" on page 295 for details about the *.bam files.

The Integrative Genomics Viewer (IGV) available from the Broad Institute, is a visualization tool for interactive exploration of large, integrated data sets. The IGV reads *.bam files directly, allowing for easy viewing and inspection of alignments against the genome.

See Figure 104 on page 300 for an example of a *.bam file visualized in IGV.

For more information, go to **www.broadinstitute.org**

The UCSC Genome Browser serves as an interactive web-based microscope that allows researchers to view all 23 chromosomes of the human genome at any scale, from a full chromosome down to an individual nucleotide.

For more information, go to **cbse.ucsc.edu/research/browser**

Saving input data and tool results files

The following list provides general guidelines about saving input and results files.

- Save the sequence data files from the instrument in the results directory. The sequence data files are the *.csfasta files and the *.qual files (color quality values).
- Save additional reports, such as the Scan_Summary.log from the Images directory
- Save the reports with Satays, color balance, and cycle heat map from the SETS export directory.
- Save files from the data/results directory including:
  - color-call_summary folder
  - run definition file
  - various plots
  - multiplexing assignment reports and more.

2 Installing BioScope™ Software

This chapter covers:

Introduction

This section describes the BioScope™ Software installation options, and system software and hardware requirements. The section also includes high-level BioScope™ Software installation instructions. For specific details, contact your Life Technologies account representative.

Installation options

Command line

This option installs all BioScope™ Software components that you access through a command line interface. Using the command line interface, you can build configuration files that can be used to assemble workflow tools.

Command line plus GUI

This option installs the command line option plus a Tomcat web server that provides a Graphical User Interface (GUI). You can use GUI to modify or create configuration files that can be used to assemble workflow tools.

Full installation

This option installs the command line, GUI, and the auto-export feature. Auto-export allows BioScope™ Software to automatically accept SOLiD™ system data from SOLiD™ instruments on a cycle-by-cycle basis. Receiving data automatically removes the time-consuming step of copying the data at the end of each run. For information about the auto-export feature, see Appendix D, "Auto-Export" on page 325.

As an option, you can install an examples directory. The directory provides BioScope™ Software sample applications, demonstration programs, configuration files, and more. You can install the examples directory before or after you install BioScope™ Software.

For information about the examples folder, see Appendix E, "Examples" on page 331.

Prerequisites

BioScope™ Software requires a Linux Beowulf cluster with a TORQUE or SGE resource manager, and a scheduler that can support scheduling policies and dynamic job priorities.

System requirements

Before you install BioScope™ Software, be sure that your headnode and clusters meet the following requirements:

- BioScope™ Software supports only Redhat and CentOS 4+ Distros on 64-bit platforms.
- At least 500 Gb of locally attached disk space or mapped storage to your compute nodes over Infiniband or SAN.
- At least 2 Tb available on the head node for the directory `/data/results` You can use any type of shared storage to meet this specification.

- At least 50 Gb must be available on the head node for the directory “/data/ reference”.

  Note: You can use any type of shared storage to meet this requirement.

- At least 2 Gb of RAM per core
- Java v1.6
- Perl v5.8.5
- Python v2.3+
- JMS v5.3+
- PBS/Torque v2.4.0+, or SGE v6.1.

  Note: If you install or use SGE, you must install “smp”

- Tomcat v6.0.18+.

If you plan to install the full version of BioScope™ Software, your cluster must include the following software and services:

- Database Postgres v8.3.6+
- Rsync v.3.0+
- Hades
- Port 8080 for Tomcat
- Port 5432 for Postgres
- On SGE clusters, a smp parallel environment

If you plan to install the CLI or the CLI plus GUI installation options, you must pre-install Java, Perl and Python on head node and on all compute nodes before you can install BioScope™ Software on the head node.

If you plan to install the full version of BioScope™ Software, you must pre-install Perl and Python and on all compute nodes before you can install BioScope™ Software on the head node.

Required services

- Java Messenger Service (JMS) JMS is sometimes referred to as “Active mq” (all installation options)
- PBS/Torque v2.4.0+, or SGE v6.1 scheduler

Plan the migration

If you are migrating from BioScope™ Software v1.1 or earlier, be sure that you delete or disable the older setup script `/etc/profile.d/corona.sh` to avoid conflict with `/etc/profile.d/bioscope_profile.sh`.

Install BioScope™ Software

This section provides typical general procedures that explain how to install BioScope™ Software. The actual procedures for your site might vary, depending on the configuration of the BioScope™ Software cluster and the installation options selected. Follow steps 1-12 for all installation options.

1. Login to **solidsoftwaretools.com/gf/project/bioscope.** If you do not have an account, contact your Applied Biosystems account representative.
2. Download `BioScope-1.2.1.tar.gz` and `BioScope-1.2.1.examples.tar.gz` to the head node.
3. Connect to the cluster.
4. Create an account called “bioscope”.
5. Add user “bioscope” to the users group.
6. Change to the directory on the head node where you copied the `*.tar.gz` files.
7. Copy the `BioScope-1.2.1.tar.gz` file to the “bioscope” home directory.
8. Enter `chown` to change the owners of the `*.tar` files to “bioscope.users”.
9. Create a directory called scratch on the compute nodes.
10. At a command prompt, enter `tar -xvzf BioScope-1.2.1.tar.gz` to untar the BioScope™ Software installation image.
11. Enter `cd Bioscope_1.2.1-installer/`.
12. Run the `install.sh` script and select the preferred options at the prompt.

Install the examples directory

1. Go to the directory where you want to install the examples folder.
2. Copy the `BioScope-1.2.1.examples.tar.gz` file to the directory where you want to install the examples folder.
3. At a command prompt, enter `tar -xvzf BioScope-1.2.1.examples.tar.gz` to untar the examples image.
4. To install the examples, run the `install.sh` script and select the preferred options at the prompt.

3 Before you Begin

This chapter covers:

Overview

This section provides general prerequisites that you might need to complete after installation and before you run BioScope™ Software, depending on the BioScope™ Software cluster configuration.

Some pre-requisite procedures described in this section require that you:

- Know the IP address of the BioScope™ Software cluster
- Have a login ID on the BioScope™ Software cluster
- Know how to:
  - navigate to directories in a Linux environment
  - edit and save files in a text editor
  - run Linux commands such as ps, pwd, cd, echo and grep

Set the BioScope™ Software environment

In some systems, the system administrator who manages the Linux server where BioScope™ Software is installed might run a script that automatically updates the .profile file with BioScope™ Software. To see if your .profile is updated, log in to the cluster and run:

```
echo $BIOSCOPEROOT
```

If the system does not display a value, make sure that you are logged in with a user ID that has write permissions on the directory that contains the .profile and perform the following steps:

1. Log in to the BioScope™ Software cluster and navigate to your home directory.

2. Modify your profile, depending on the shell:
   - bash - .bashrc_profile, .bashrc, or .profile
   - csh/tcsh - .cshrc
   - Korn - .profile

3. Set the BioScope™ Software path:

```
# Check if BISOCOPEROOT is set and set it if null
: ${BIOSCOPEROOT:=/share/apps/bioscope}
export BIOSCOPEROOT

if [ -d ${BIOSCOPEROOT}/etc/profile.d ]
then
   for i in ${BIOSCOPEROOT}/etc/profile.d/*.sh; do
      if [ -r "$i" ]; then
          . $i
      fi
   done
   unset i
fi
Run the script generate-profile.csh to set the C shell
environment path:
```

```
# Check if BIOSCOPEROOT is set and set it if null
if ( ! ${?BIOSCOPEROOT} ) then
        setenv BIOSCOPEROOT /share/apps/bioscope
endif

if ( -d ${BIOSCOPEROOT}/etc/profile.d ) then
        set nonomatch
        foreach i ( ${BIOSCOPEROOT}/etc/profile.d/*.csh )
                if ( -r $i ) then
                        source $i
                endif
        end
        unset i nonomatch
endif
```

4. Verify the setup by displaying the path. Enter:

```
echo $BIOSCOPEROOT
```

If the command does not return a value, the setup is incorrect. Contact your system administrator.

Start the Java Messenger Service (JMS)

BioScope™ Software supports running only one session of JMS. Once started, JMS is available to all users.

To see if JMS is running, enter:

```
ps -elf | grep activemq
```

If the system returns information that JMS is running in multiple instances, ask your system administrator to stop all but one JMS instance.

If the system returns information that JMS is not running, complete the following steps:

- If your cluster is configured with the full-install option of BioScope™ Software, enter:

  ```
  service activemqd start
  ```

- If your cluster is configured with the Command and GUI option of BioScope™ Software, change to the directory where BioScope™ Software is installed and enter:

  ```
  ./start-ActiveMQ.sh
  ```

Create the directory structure

The BioScope™ Software configuration files are a set of key-value pairs. In general, keys correspond to command line parameters and values are what you would typically set at the command prompt. These keys follow the syntax used by Java Properties files. Text up to the equal sign is the key and text from the equal sign to the new line is the key value. The BioScope™ Software configuration files are enhanced beyond a one-to-one parameter mapping in a number of ways. Key values can combine or extend the values of other keys using a special quoting syntax. This greatly simplifies setting common directory roots, or other derived values.

```
# BioScope key value pairs

# key value pair that defines a general output
directory

output.dir = /data/results/output

# key value pair that uses the output.dir
variable described above

mapping.output.dir = ${output.dir}/mapping
```

Another valuable feature is the import statement. During runtime, this statement takes a path to another configuration file in which the contents are imported into the current *.ini file. The import feature helps ensure that parameters are relatively constant.

For example, the second line of the diBayes.ini file contains the command:

```
    import ../<globals>/global.ini
```

An example of the global.ini file is shown in the following section:

```
############################
############################
##
##  global parameters
##
base.dir=./
output.dir = ${base.dir}/outputs
temp.dir = ${base.dir}/temp
intermediate.dir = ${base.dir}/intermediate
log.dir = ${base.dir}/log
reads.result.dir.1 = ${base.dir}/reads/F3
reads.result.dir.2 = ${base.dir}/reads/R3
reference.dir = /data/results/bioscope1.2/examples/demos/
references/
reference = ${reference.dir}/hg18_validated.fasta
scratch.dir=/scratch/solid
```

To further aid in the development of more generic configuration files, some predefined keys are available. For example, the `bioscope.start.dir` key has a value equal to the directory in which a BioScope™ Software analysis is started. This supports the development of configurations with relative locations that can be used for various types of analyses.

Set up tool directories

BioScope™ Software tools recognize a common set of directory types (see Table 4).

Note: If you selected the Full Installation option, most directories are created during installation time. If you selected the CLI or CLI and GUI installation options, you must create the directories after installing BioScope™ Software.

If you installed the examples directory, you can populate your directories with the sample data provided in the examples directory. The examples/plugins directory contains samples of all the BioScope™ Software *.ini files. For information about the examples directory, see "Examples" on page 331.

Table 4 Tool results directory folder description

Directory type	Description	General key
Output	Output directory for end user files.	output.dir
Shared temp	Directory for files that will be cleaned up after an analysis is done.	tmp.dir
Node-local temporary (scratch)	In clustered environments, it is often useful to use the local disk for I/O-intensive work. This key specifies the directory use. It must be possible to create this directory on the local disk.	scratch.dir
Intermediate	Some BioScope™ Software tools perform resource- intensive preparations prior to analyze the data. You can use the intermediate directory to store files for reuse when analysis parameters only are changed.	intermediate.dir

For maximum flexibility, each tool has a key that corresponds to the directory types relevant for the tool, for example, the mapping.output.dir parameter.

Except for scratch, these directories are typically defined underneath a common root that corresponds to the unit of analysis. For secondary analysis this is usually a slide.

```
# F3 mapping.ini

slide.dir = /data/results/solid_slide_1

output.dir = ${slide.dir}/output

tmp.dir = ${slide.dir}/tmp

intermediate.dir= ${slide.dir}/intermediate

mapping.run = 1

mapping.output.dir = ${output.dir}/F3_mapping

mapping.tmp.dir = ${tmp.dir}/F3_mapping
```

```
# R3 mapping.ini
slide.dir = /data/results/solid_slide_1
output.dir = ${slide.dir}/output
tmp.dir = ${slide.dir}/tmp
intermediate.dir= ${slide.dir}/intermediate
mapping.run = 1
mapping.output.dir = ${output.dir}/R3_mapping
mapping.tmp.dir = ${tmp.dir}/R3_mapping
```

Scratch is set up based on a cluster configuration. Unless there is no node-local storage available or you are troubleshooting, settings in the installation property files should be correct.

Having a design strategy makes file maintenance easier. When you plan to run multiple slides for a single analysis, it is a good idea to set the directory types under common parent directories. The example above shows the temporary directories from both F3 and R3 mapping under the same parent (`/data/results/solid_slide_1/tmp`). When organized this way, it is easy to remove unnecessary files after multiple tools have been run.

Set up tertiary analysis directories

A tertiary analysis, such as SNP detection and identification of structural variation, is set up the same way as a secondary analysis. Tertiary analyses use configuration files and standard keys. However, input for tertiary analysis often comes from multiple slides and is set up outside the secondary results tree.

For example, you might design a tertiary results tree that is similar to:

```
/data/results/tertiary/output
/data/results/tertiary/intermediate
```

If the BioScope™ Software examples directory is installed on the cluster, you can copy the examples of the tertiary *.ini and related files to their respective directories. You can customize the example files to fit your configuration. For more information about the BioScope™ Software examples directory, see Appendix E, "Examples" on page 331.

The following example shows the cnv.ini file, which is a tertiary configuration file:

```
##INI BEGIN
################################################################
####
#    Global Settings for CNV Tool
################################################################
####
base.dir = .

# Switch to turn it on (1) or off (0)
cnv.run = 1

##############################
```

```
# Mandatory parameters
##############################
# Experiment Name
experiment.name = Test_run
# Output directory
cnv.output.dir = ${base.dir}/outputs
# Log directory
cnv.log.dir = ${base.dir}/log-dir
# Intermediate directory
cnv.intermediate.dir = ${base.dir}/intermediate-dir
# Coverage format [GFF|Binary]
coverage.format = GFF
# Inputs in the format
# coverage.file.info=data-type:tag-length:coverage-
format:coverage-file,...
# For example, for coverage-format Binary:
# coverage.file.info = /data/cnvrun/inputs/inputCoverageFiles
# For coverage-format GFF:
# coverage.file.info = /data/cnvrun/inputs/GFF/GFF1.gff,/data/
cnvrun/inputs/GFF/GFF2.gff
coverage.file.info = ${base.dir}/gff3_input_chr1/
human_chr1_gff3.gff
# Path to the CMAP file
cmap.file = ${base.dir}/cmap/referenceMapping.cmap
##INI END
```

Prepare the reference file

You must perform three procedures on your reference file before you can use it with a tool:

1. Validate the reference file to a format that complies with BioScope™ Software.
2. Create the reference.properties file.
3. Concatenate the reference file.

IMPORTANT! Do not validate the reference file that you download for use with the Human CNV tool. See Chapter 12, "Run the Find Human CNVs Tool" on page 201 for more information.

Validate the reference file

Reference validation is not performed automatically. You must validate the reference file so that it is presented in the correct format to each tool that uses a reference file.

Usage parameters

```
-r <reference_file>
-s <reference_size_limit>
-o <outputfile>
```

Validation procedure

1. Login to the BioScope™ Software cluster. Be sure that you log in with a user name that has “x” privileges on the directory that contains the reference_validation.pl script.

2. The script is under the bin folder where the BioScope™ Software is installed.

3. Navigate to the directory that contains the reference_validation.pl file.

4. At a command prompt, enter:

   ```
   $BIOSCOPEROOT/bin/reference_validation.pl <usage parameters>
   ```

Create the reference.properties file

The BioScope™ Software mapping and pairing pipelines will try to create and read the properties file in the same directory as the reference file by default. You can specify any directory via the reference.properties.dir parameter.

In general, try to run the validation script on the reference file before you run the script to create the reference.properties file. The purpose of the references.properties file is to make a summary of key contents of the reference file, so that the mapping tool does not have to read every line in the reference file. The reference properties script is installed with the BioScope™ Software application. If you do not create a reference.properties file before beginning a mapping run, the mapping and pairing tools will create a reference.properties file and attempt to write the reference.properties.file to the directory where the reference file resides.

If you start the mapping and pairing run with a user id that does not have write privileges to the directory where the reference file is stored, BioScope™ Software will not be able to store the reference.properties file in the reference directory and the program will produce an error.

The following list provides general guidelines to follow when working with the reference.properties file:

- Option 1: Create the references.properties file before running mapping and store the references.properties file in a directory.
- Option 2: update globals.ini with a key/value pair that points to the directory where references.properties is stored. The parameter name is `reference.properties.dir`
- You must set your environment before running the properties reference script (see “Set the BioScope™ Software environment” on page 36)

Concatenate the reference file

Perform this step to concatenate two individual chromosome files, such as files you might have if you download the hg18 genome.

1. Log in to the BioScope™ Software cluster.

2. At a command prompt, enter:

```
$ cat chr1.fa chr2.fa chr3.fa ... chrM.fa > hg18.concatenated.fa
```

The ellipsis clips out the other chromosome files.

Convert the .gtf file

Not all species have exactly the same format of GTF file. Before using a *.gtf file from a public source with a BioScope™ Software tool, you must convert the *.gtf file into a format that is compatible with BioScope™ Software.

Convert the Ensembl gtf file

The ENSEMBL website **www.ensembl.org/** has *.gtf-formatted genome annotations available for many popular assemblies. ENSEMBL *.gtf files are properly normalized by gene and transcript IDs.index.html

ENSEMBL *.gtf files use gene accession numbers instead of HUGO-style gene names. ENSEMBL *.gtf files also use unprefixed sequence identifiers, such as 1,2,3….X,Y,MT. The ENSEMBL *.gtf files are incompatible with genome reference *.fasta files that have UCSC-style sequence IDs with the prefix "chr", for example, chr1, chr2, chr3….chrX, chrY, chrM.

Reformat the ENSEMBL .*gtf file

1. Login to the BioScope™ Software cluster.
2. Run `convert_ensembl_gtf.pl`:

```
% reformat_ensembl_gtf.pl Homo_sapiens.GR
```

Convert the refGene.txt.gz file

The UCSC Genome Browser has genome annotations available for many assemblies at

hgdownload.cse.ucsc.edu/goldenPath/

The *.gtf-formatted annotations available for download are not properly normalized by gene ID. The required content is present for each assembly in the file export of the *refGene database table database/refGene.txt.gz*

For example, annotation for human genome build 18 is available at:

hgdownload.cse.ucsc.edu/goldenPath/hg18/database/refGene.txt.gz

Note: The *.gtf-formatted annotation is not in *.gtf format. You must convert the annotation before using it in WTA.

Run the script `bin/refgene2gtf.sh` to convert the refGene.txt.gz file:

```
% gunzip refGene.txt.gz
```

```
% refgene2gtf.sh -i refGene.txt -o refGene.gtf
```

Genome annotations that are downloaded from the UCSC Genome Browser and converted by the annotation conversion script are optimal because they contain Human Genome Organization (HUGO)-style gene names. HUGO-style gene names allow interpretation when using a genome browser or reading reports.

The annotation conversion script works with the latest format of `refGene.txt` files. Assemblies, such as the rat genome, use an alternative format for the `refGene.txt` file. The `refgene2gtf.sh` script does not convert alternative formats.

Update the global.ini file

1. Login to the BioScope™ Software cluster.
2. Navigate to the plugins directory in the examples folder.
3. Open global.ini.
4. Update the path to the "scratch" and "results" directories.
5. Save the global.ini file.

Run the SAET color correction tool

Optional. See Appendix B, "Use the SOLiD™ 4 Accuracy Enhancer Tool" on page 311 for details about running the SAET color correction tool.

Verify the browser version

BioScope™ Software supports:

- Internet Explorer® versions 6 and 7
- Mozilla® 3.0.1

Verify queue availability

Verify that the `bs_primary` and `bs_secondary` queues are available.

Run the experiments in the examples directory

The experiments in "/examples/demos" contain all of the files required to perform a sample experiment, such as mapping.

See Appendix E, "Examples" on page 331 for details about the BioScope™ Software examples.

Perform the stress test

For information about performing the stress test, contact your Life Technology account representative.

Verify libraries for readbuilds and autoexport

Verify that the libraries necessary for readbuilder and autoexport are present on the BioScope™ Software cluster. The autoexport feature is only available if the Full Installation option was selected.

- libactivemq-cpp-3.0.1
- libapr-1.3.6
- libaprutil-1.3.8
- libhdf5-1.8.3
- libboost-1.36.0
- libxerces-c-2.8
- hdf5tools-1.8.3

Review supported library types

Table 5 lists the library types supported by each BioScope™ pipeline.

Table 5 Supported library types

Tool	Library type		
	Fragment	**Paired-end**	**Mate-pair**
Resequencing			
Mapping	Yes	Yes	Yes
Human CNV	Yes	Yes	Yes
Inversion	No	No	Yes
SNP Finding	Yes	Yes	Yes
Large Indel	No	Yes	Yes
Small Indel	Yes	Yes	Yes
Whole Transcriptome Analysis			
Coverage	Yes	Yes	Yes
Known Exons	Yes	Yes	Not applicable
Splice junctions	No	Yes	Not applicable
Gene Fusions	No	Yes	Not applicable

4 Whole Transcriptome Pipeline Concepts

This chapter contains:

Purpose

The chapter has three purposes:

- To provide instructions for running whole transcriptome analysis (WTA) single-read and paired-end pipelines using BioScope™ Software.
- To list and describe configurable BioScope™ Software parameters.
- To provide information about the underlying algorithms that power BioScope™ Software tools.

Background

You can use the SOLiD™ 4 system to sequence RNA prepared with RNA-Seq sample preparation kits. RNA sequencing produces high depth, short read sequencing data that can be used to measure RNA expression. Like microarray analysis, RNA-Seq measures expression intensity across many genomic features. Unlike microarray analysis, RNA-Seq can be used to identify novel transcriptome features in a sample.

Overview

Like other SOLiD™ 4 applications, RNA-Seq produces short reads in the form of *.csfasta and *.qual files. Using BioScope™ Software, WTA pipelines take these reads as input, and perform the following steps:

1. Align reads - The aligning reads step identifies the alignments between the reads and the reference genome sequence and reports the alignments as a *.bam file.

2. Count known exons - The counting known exons step identifies the number of reads that align within genomic features

3. Calculate coverage - The calculating coverage step reports the read coverage at each genomic position.

4. Finding junctions - The finding junctions step identifies splice junctions of various types, including fusion junctions from paired-end reads.

Finding junctions

The SOLiD™ 4 system produces RNA-Seq reads in both single-read and paired-end configurations. While there is overlap in the analytical techniques, the analysis of single-read and paired-end data is performed with two distinct pipelines. The single-read pipeline consists of alignment, counting, and coverage steps. The paired-end pipeline extends single-read analysis with an alternative alignment step and an additional junction finding step.

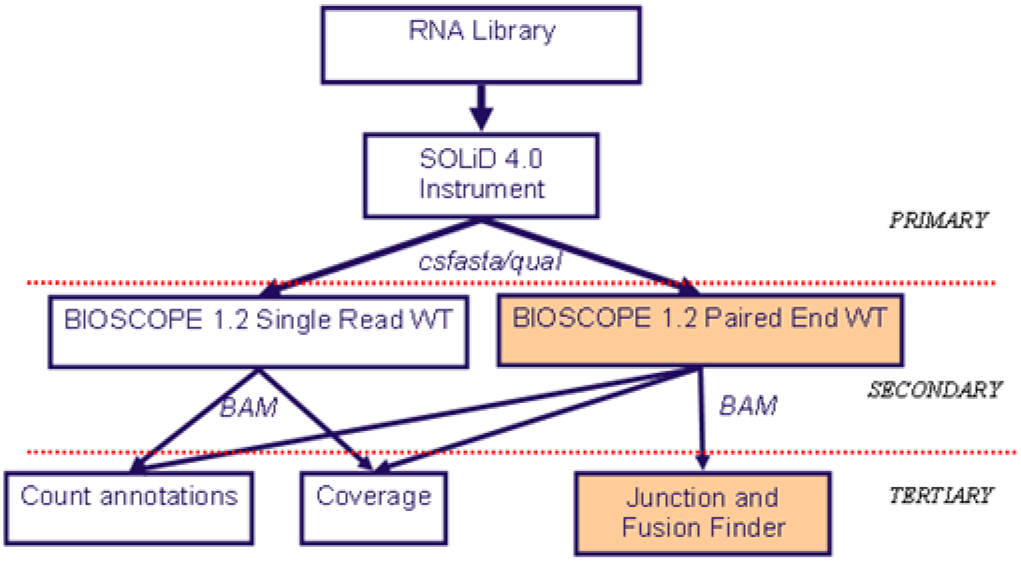


Figure 5 WTA single-read and paired-end pipeline workflows

The following sections of this document provide background information about secondary analysis (mapping and pairing, also known as aligning reads) and tertiary analysis (data visualization and analysis, including counting tags, determining coverage, and finding junctions). The information in these sections includes descriptions of the parameters you may want to configure to customize your analysis.

Secondary analysis - aligning reads

The RNA-Seq reads used in WTA are different from the genomic reads. For RNA-Seq reads:

- Only transcribed sequences are measured by the system.
- Genome coverage is non-uniform due to variation in transcriptional intensity.
- A large subset of reads originate from “uninteresting” sequences such as ribosomal RNA (rRNA).
- A subset of the reads originates from splice junctions and cannot align contiguously on the genome.

While genomic resequencing relies only on alignment to a reference sequence, WTA also makes use of gene annotations. Gene annotations define the exons, genes, and transcripts used to improve alignment.

In spite of their differences, single-read and paired-end pipelines share several components. In the following sections of this document, these individual components are described before the pipelines.

Mapping reads

The first steps of WT mapping and alignment are performed in the same way as resequencing mapping and alignment. In step one, WT mapping uses the read mapping feature of BioScope™ Software. In step two, reads are mapped to the entire genomic reference.

After the two mapping steps are complete, there are additional WT-specific mapping steps:

- Filter mapping
- Junction mapping
- Exon mapping

Each mapping step is implemented as a separate instance of the BioScope™ Software mapping feature within a WT pipeline. The filter, junction, and exon mapping steps are distinguished by the reference sequence used. The filter, junction, and exon mapping steps are configured identically for each set of reads.

Filter mapping

The reads are mapped to the filter reference in the filter mapping step. Reads that align to the filter reference are called filtered reads. The result of this mapping step is used to populate the filter report. Filtered reads are annotated in the single-read *.bam file; Filtered reads are omitted from the paired-end *.bam file. A filter reference for use with human reads is provided with BioScope™ Software. This reference contains:

- SOLiD™ 4 adapter sequences
- Human ribosomal RNAs (rRNAs)
- Human transfer RNAs (tRNAs)
- Other human sequences
- Single-base-repeat sequences, including Poly-A, Poly-T, and more

Complete the following steps to construct a filter reference for another species.

1. Copy the adapter and single-base-repeat sequences to a new file.

2. Append *.fasta-formatted species-specific ribosomal, transfer RNA, and other sequences to the filter reference.

Junction mapping

Reads are mapped to the flanking sequence of junctions defined by the genome annotation in the junction mapping step. Junctions are inferred from the exon, gene, and transcript definitions in the genome annotation. Junction sequences are produced for all splices between pairs of exons within a gene. Junctions present in annotated transcripts are called known junctions. Junctions not present in annotated transcripts are called putative junctions.

Exon mapping

In the exon mapping step, reads are mapped to the set of exon sequences defined by the genome annotation. This step maps the shorter F5 reads in the paired-end pipeline. By default, more mismatches are allowed in exon mapping of F5 reads in the paired-end pipeline.

Table 6 describes a human exon sequence database and a human junctions database.

Table 6 Comparison of alternative human references

	Reference	Number of references	bp size	Mappability	Novelty	Junctions
F3	Genome	23	3 billion	OK	OK	No
	Junctions	2,012,075	201,207,500	OK	No	OK
	Refseq	32,699	99,02,749	OK	No	Partial

Table 6 Comparison of alternative human references *(continued)*

	Reference	Number of references	bp size	Mappability	Novelty	Junctions
F5	Genome	23	3 billion	Not	OK	No
	Exons	216,884	99,026,749	OK	No	No
	Refseq	32,699	99,026,749	OK	No	Partial

WTA single-read alignment pipeline

This section provides information about workflow, parameters and output files for the single-read pipeline (see Figure 6 on page 52).

Workflow

In a single-read alignment, reads are mapped to the filter, genome, and junction sequences. The results of these distinct mapping steps are merged and output in the form of a single-indexed *.bam file as well as reports characterizing alignments to the filter and genome reference.

Alignments of a read to a genome and to junctions can produce multiple similar alignments at the same location. When multiple similar alignments occur, the alignment with the highest alignment score replaces all others. If there is a tie, the genomic alignment is used. The alignment score is calculated as follows:

$$Score = len - nm \times 1(1 + mp) - jp$$

where:

- *len* = number of colors in the alignment
- *nm* = number of color mismatches
- *mp* = mapping mismatch penalty
- *jp* = penalty for alignment to a junction

An accurate mapping quality for alignments spanning splice junctions has not been developed for single-read WTA. Therefore, in the single-read pipeline, the mapping quality of a junction-alignment is set as follows:

- If the junction alignment has an ungapped genomic alignment counterpart, the junction alignment takes the mapping quality of the contiguously mapped alignment.
- If the junction alignment has no such counterpart, the mapping quality is set to 255, which is the unknown mapping quality.

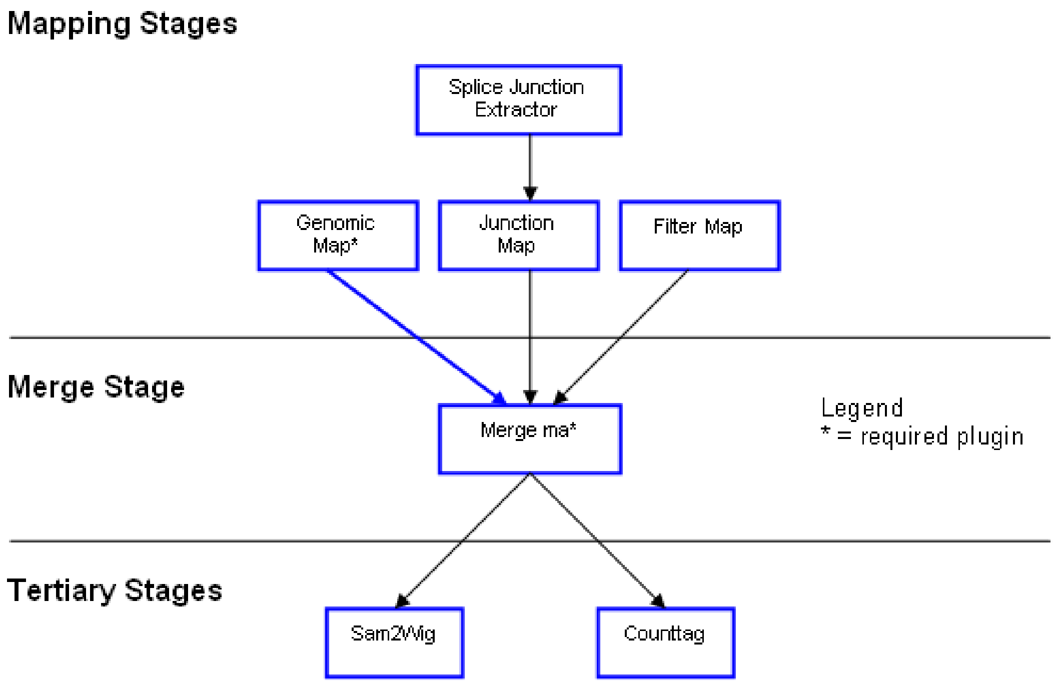


Figure 6 WT single-read pipeline flow

Single-read WTA parameters

Table 7 describes the algorithm parameters that you can configure for single-read WTA. Table 8 on page 53 describes the input, output, and parallelization parameters that you can configure for single-read WTA.

Table 7 Single-read WTA algorithm parameters

Parameter	Default	Description
wt.spljunctionextractor.run	1	Enables splice junction extraction, which is required for junction mapping.
wt.junction.mapping.run	1	Enables junction mapping.
wt.genomic.mapping.run	1	Enables genomic mapping.
wt.merge.run	1	Enables the step that merges individual mapping results, writes the *.bam file.
wt.merge.known.junction.penalty	0	Penalty applied to the score of alignments spanning known splice junctions.
wt.merge.putative.junction.penalty	1	Penalty applied to the score of alignments spanning putative splice junctions.
wt.merge.score.clear.zone	5	Used to identify unique reads for stats in the alignment report. Has no effect on the *.bam file.
wt.merge.min.junction.overhang	8	Minimum number of bases a read must match on both sides of a junction in order for a junction alignment to be reported.

Table 7 Single-read WTA algorithm parameters *(continued)*

Parameter	Default	Description
wt.merge.num.alignments.to.store	2	Maximum number of alignments to report per read.
read.length	50	Length of the reads.
wt.genomic.mapping.scheme.unmapped.25 wt.junction.mapping.scheme.unmapped.25 wt.filter.mapping.scheme.unmapped.25 wt.genomic.mapping.scheme.repetetive.25 wt.junction.mapping.scheme.repetetive.25 wt.filter.mapping.scheme.repetetive.25	25.2.0:10 (unmapped) <none> (repetitive)	Mapping scheme that is used to map 25 color reads to the genomic, junction and filter references.
wt.genomic.mapping.scheme.unmapped.35 wt.junction.mapping.scheme.unmapped.35 wt.filter.mapping.scheme.unmapped.35 wt.genomic.mapping.scheme.repetetive.35 wt.junction.mapping.scheme.repetetive.35 wt.filter.mapping.scheme.repetetive 35	35 25.2.0:10 (unmapped) <none> (repetitive)	Mapping scheme used for 35 color reads to the genome, junction, and filter references.
wt.genomic.mapping.scheme.unmapped.50 wt.junction.mapping.scheme.unmapped.50 wt.filter.mapping.scheme.unmapped.50 wt.genomic.mapping.scheme.repetetive.50 wt.junction.mapping.scheme.repetetive.50 wt.filter.mapping.scheme.repetetive.50	25.2.0:20 (unmapped) <none> (repetitive)	Mapping scheme used for 50 color reads to the genome, junction, and filter references

Table 8 Single-read WTA input, output, and parallelization parameters

Parameter	Value	Description
Input parameters		
reference.file	–	The .fasta file that contains the genomic reference sequences.
exons.gtf.file	–	The .gtf file that defines the exons, transcripts, and genes that create the junction sequences.
mapping.tagfiles	–	The *.csfasta file that contains the color-space reads.
qual.file	–	The *.qual file that contains the quality values of the color-space reads.
Output parameters		
merge.output.directory	output/single_read/mapping	Directory containing all alignment output.
merge.output.bam.file		Name of the *.bam file created during mapping and pairing.

Table 8 Single-read WTA input, output, and parallelization parameters *(continued)*

Parameter	Value	Description
Parallelization parameters		
mapping.number.of.nodes	7	The number of nodes used for mapping jobs.
mapping.np.per.node	8	The number of processors per node used for mapping jobs.
mapping.min.reads	10,000,000	The minimum number of reads for mapping jobs to be distributed on different nodes.
mapping.memory.size	–	The amount of memory used for mapping.

Single-read output files

A *.bam file produced by the WT paired-end pipeline is identical to that produced by the resequencing pipeline. However, the *.bam format from the WT-single-read pipeline differs from *.bam files produced elsewhere in BioScope™ Software. The single read pipeline produces separate *.bam files for filtered, unmapped and mapped reads.

Note: The WTA single-read pipeline produces a *.bam file that uses the optional fields described in Table 9.

Table 9 WT single-read pipeline *.bam file optional fields

Optional field	Description
IH:i:	Number of stored alignments containing the current query.
HI:i:	Query hit index.
NH:i:	Number of reported alignments containing the current query.
CS:z:	Color read sequence.
CQ:z:	Color quality sequence.
CC:z:	Reference name of the next hit.
CP:z:	Coordinate of the next hit.
AS:i:	locationAlignment Score generated by the aligner.
XN:i:	Alignment score of the best non primary alignment for query in the current record.
XF:z:	T for true or F for false. Set to "T" if this read is filtered.
XJ:z:	"K" for Known Junction or "P" for Putative Junction.

Paired-end alignment pipeline

This section provides information about the paired-end alignment pipeline workflow, mapping reads, and pairing reads (see Figure 7).

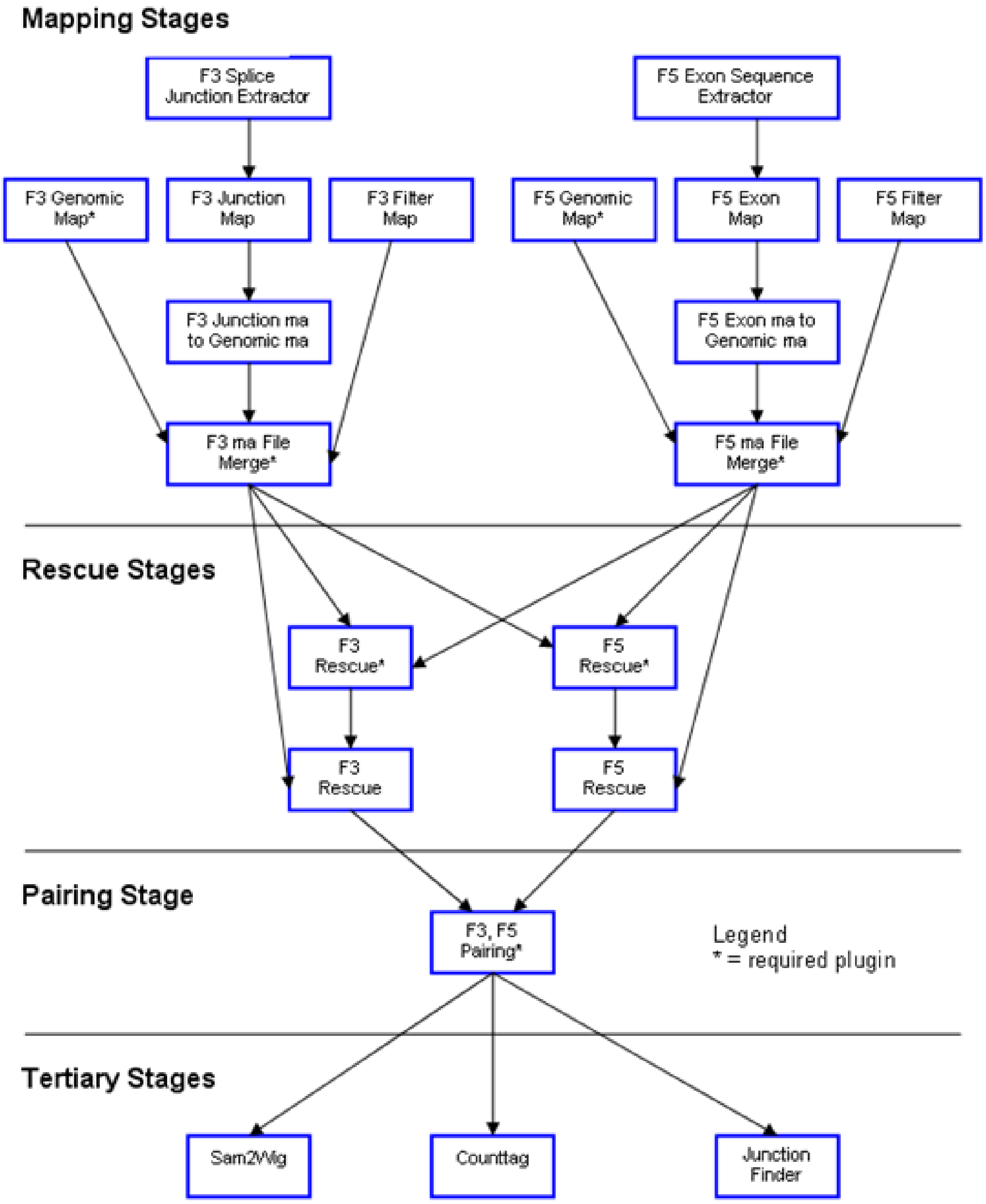


Figure 7 Paired-end mapping workflow

Workflow

A paired-end alignment, like a single-read alignment, consists of mapping and pairing steps. In paired-end alignment, mapping is the alignment of individual reads to the reference. Pairing is the processing of read pairs, using the pairing information to refine the alignments.

Mapping reads

In a paired-end alignment, the F3 and F5 paired-ends are initially processed through independent mapping paths. Similar to single-read-mapping, the F3 reads are mapped to filter, genome, and junction sequences. The F5 reads are also mapped to filter, genome, and exon sequences. Reads that map to filter sequences are discarded and do not proceed further in the pipeline.

Note: Reads that map to filter sequences in a paired-end alignment are discarded and are not part of the *.bam file. The single-read pipeline includes filtered reads in the *.bam file.

The F3 genomic and junction mapping results are merged into a nonredundant set of alignments. The F5 genomic and exon mappings are merged into a non-redundant set of alignments. At this point in the analysis, the output consists of a set of independently processed F3 and F5 genomic alignments.

Rescue method

You can use the rescue method to find additional alignments. Rescue is an alignment method that is applied to read pairs that have at least one alignment, but no pair of alignments occurring within an expected range (see Figure 8 on page 56). The expected range is set to 100,000 bases by default.

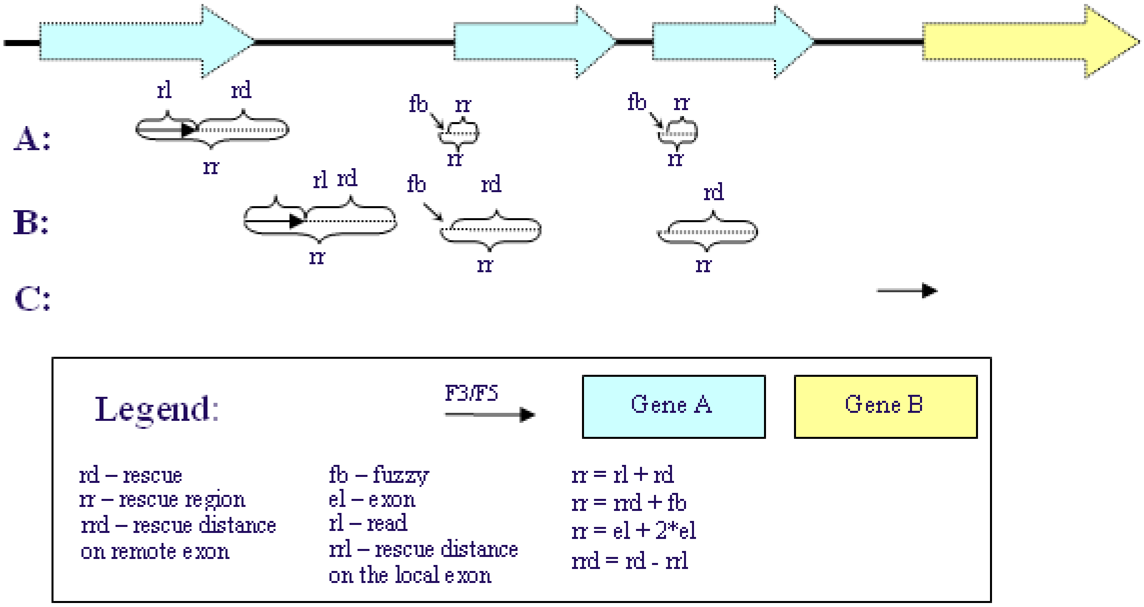


Figure 8 Annotation-aided rescue for WT

The next section refers to Figure 8.

Row A

For alignments that fall on an exon of a gene, and do not have a mate alignment above a certain score threshold, a special exon rescue is performed. Rescue region (rr) is defined as a rescue distance that is downstream of the alignment. The rescue distance is determined by user-defined thresholds. The rescue distance starts from the left-most position of the alignment for overlapping mates. The rescue region also includes certain position range in downstream exons of the same gene. The approach defined in the previous sentences helps rescue mates on different exons.

Row B

A rescue is still performed on downstream exons if a read is mapped on the intron of a given gene.

Row C

No special exon rescue is performed if an alignment falls on intergenic region. Regular rescue within pairing distance is performed.

Alignments that lack a sibling read aligned nearby are called anchors. If anchors are found, the rescue tool conducts a more sensitive search for the sibling near the anchor, within a limited region of the genome. In the WT paired-end pipeline, the search is limited to anchors that occur within the introns or exons of annotated genes, with an allowance for a few overhanging bases.

Rescue is performed only within a set of expected rescue distances determined by a gene's exon structure and the insert size distribution. Rescue distances are a function of the transcript rescue distance:

transcript rescue distance = mean insert size + 3 standard deviations

This distance is longer than the majority (99.7%) of inserts. The formula above describes the rescue distance without taking into account the presence of introns. Because inserts are very likely to contain introns, a rescue distance is calculated for each potential splice configuration of the gene. For each configuration, the rescue distance is the transcript rescue distance + length of introns within the relevant transcribed sequence interval. Alternatively, rescue can be performed on all exons to improve sensitivity, but with a substantial increase in false positives and run time. Rescue is optional and can be applied using F3 anchors only, F5 anchors only, or both.

Regardless of the rescue steps employed, the rescue results supplement the alignments detected in individual mapping steps. The end result of rescue is a set of F3 and F5 alignments in the same format as that produced by mapping.

Pairing reads

In the pairing step, pairs of reads are evaluated, assigned a mapping quality value, and written to a *.bam file. The pairing range is set to 100,000 so that reads in adjacent exons are tagged as proper pairs. Unlike pairing of genomic resequencing data, the mapping quality of read pairs is a function of genome annotation. Alignment pairs that do not occur within the same gene are penalized. For each pair, the alignment with the highest quality value is designated as the primary alignment. In the case of multiple highest quality, a single-pair alignment is selected randomly. Figure 9 shows an example of pairing range calculation with junction alignments.

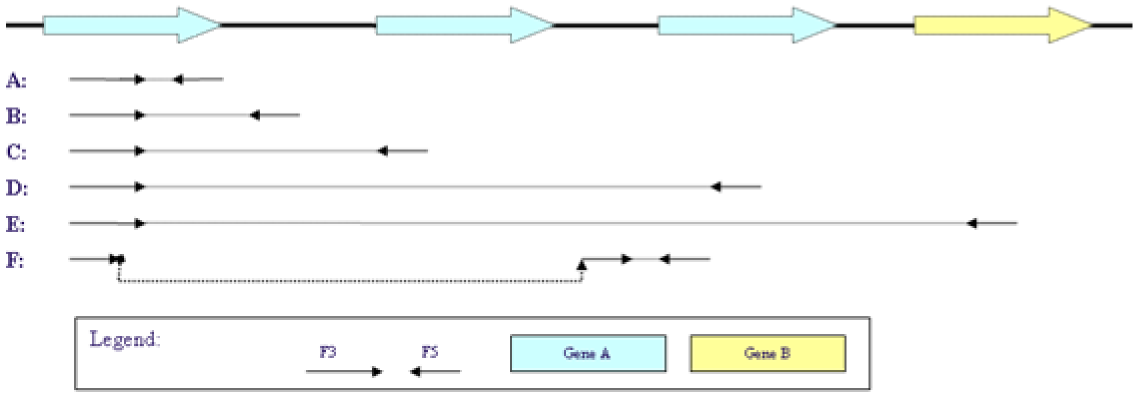


Figure 9 WT annotation-aided alignment Pairing Quality Value (PQV)

See Table 10 for details about rows A to F in Figure 9.

Table 10 Annotation-aided alignment PQV description

Row(s)	Description	PVQ penalized?
A	Mates fall on the same exon	No.
B	One mate falls on an exon and the other falls on an intron	Yes.
C and D	Mates fall on separate exons of the same gene	No.
E	Mates fall on exons of different genes	Yes
F	Spliced alignment where one mate partially falls on a known gene and the other falls on the exon of the same gene	No

Parameters

Table 11 describes the algorithm parameters that you can configure for paired-end WTA. Figure 8 on page 53 describes the input, output, and parallelization parameters that you can configure for single-read WTA. Table 12 on page 60 describes the input and output parameters for paired-end WTA.

Table 11 Paired-end WTA algorithm parameters

Algorithm parameter	Default	Description
wt.f3.genomic.mapping.plugin.run	1	Enables the genomic mapping of the F3 reads in the pipeline.
wt.f3.filter.mapping.plugin.run	1	Enables the filter mapping of the F3 reads in the pipeline.
wt.f3.splice.junction.extractor.plugin.run	1	Enables the splice junction extraction, a required step for junction mapping.
wt.f3.junction.mapping.plugin.run	1	Enables the junction mapping of the F3 reads in the pipeline.
wt.f3.junction.ma.genomic.ma.plugin.run	1	Enables the conversion of F3 junction mappings into genomic mappings.
wt.f3.ma.file.merger.into.ma.file.plugin.run	1	Enables the merging of genomic and junction mappings.
wt.f5.genomic.mapping.plugin.run	1	Enables the genomic mapping of the F5 reads.
wt.f5.filter.mapping.plugin.run	1	Enables filter mapping of the F5 reads.
wt.f5.exon.sequence.extractor.plugin.run	1	Enables the exon sequence extraction, which is a step required for exon mapping.
wt.f5.exon.mapping.plugin.run	1	Enables exon mapping of the F5 reads.
wt.f5.exon.ma.to.genomic.ma.plugin.run	1	Enables the conversion of F5 exon mappings to genomic mappings.
wt.f5.ma.file.merger.into.ma.file.plugin.run	1	Enables the merging of genomic and exon mappings.
wt.f3.exon.table.rescue.plugin.run	1	Enables the rescue of F3 reads.
pairing-wt.run	1	Enables pairing.

Table 11 Paired-end WTA algorithm parameters *(continued)*

Algorithm parameter	Default	Description
genomic.mapping.scheme.unmapped.25 junction.mapping.scheme.unmapped.25 exon.mapping.scheme.unmapped.25 filter.mapping.scheme.unmapped.25 genomic.mapping.scheme.repetetive.25 junction.mapping.scheme.repetetive.25 exon.mapping.scheme.repetetive.25 filter.mapping.scheme.repetetive.25	25.2.0:10 (unmapped) <none> (repetitive)	The mapping scheme used for mapping 25 color reads to the genomic, junction, and filter references.
genomic.mapping.scheme.unmapped.35 junction.mapping.scheme.unmapped.35 filter.mapping.scheme.unmapped.35 exon.mapping.scheme.unmapped.35 genomic.mapping.scheme.repetetive.35 junction.mapping.scheme.repetetive.35 exon.mapping.scheme.repetetive.35 wt.filter.mapping.scheme.repetetive.35	25.2.0:10 (unmapped) <none> (repetitive)	The mapping scheme used for 35 color reads to the genome, junction, and filter references.
genomic.mapping.scheme.unmapped.50 junction.mapping.scheme.unmapped.50 exon.mapping.scheme.unmapped.50 filter.mapping.scheme.unmapped.50 genomic.mapping.scheme.repetetive.50 junction.mapping.scheme.repetetive.50 exon.mapping.scheme.repetetive.50 filter.mapping.scheme.repetetive.50	25.2.0:20 (unmapped) <none> (repetitive)	The mapping scheme used for 50 color reads to the genome, junction and filter references.
wt.rescue.run.input.genereration	1	Turn on/off the rescue input generation
wt.rescue.run.rescue	1	Turn on/off the rescue program
wt.rescue.input.generation.avg.insert.size	120	Average insert size. Must be set if the protocol results in a different value.
wt.rescue.input.generation.std.insert.size	60	The standard deviation of the rescue input generation insert size.
wt.rescue.input.generation.rescue.only.unaligned reads	0	Only unaligned reads are rescued.
wt.rescue.input.generation.rescue.short.range	1	Leaving the default value will rescue all the target reads in the vicinity of the anchor read.
wt.rescue.input.generation.rescue.only.for.the.best.anc hor.alignment	0	Only the best alignment is used to anchor rescue.

Table 11 Paired-end WTA algorithm parameters *(continued)*

Algorithm parameter	Default	Description
wt.rescue.input.generation.rescue.fuzzy.exon.borders	1	Allow rescue in extra bases beyond exon boundaries.
wt.rescue.input.generation.rescue.anchor.alignments.not.overlapping.exons	1	If you retain the default value of 1, BioScope™ Software also rescues downstream and upstream of alignments that are not overlapping exons, but are overlapping genes.
wt.rescue.input.generation.rescue.only.within.rescue.distance	1	If you retain the default value of 1, BioScope™ Software rescues only within the rescue region starting from the exon border to a limit inside the exon. The limit is computed based on the insert size and the location of the anchor alignment inside the previous exon. If you set the value to 0, BioScope™ Software rescues on the entire exon.
wt.rescue.input.generation.exon.fuzzy.border.width	10	The number of extra bases outside exon boundaries in which to allow rescue.
wt.rescue.input.generation.min.alignment.distance.for.rescue	100,000	This value is the minimum distance between the anchor alignment and any of the alignments of the rescued read downstream of the anchor alignment when rescuing reads that have been already mapped to the reference.
wt.rescue.mask.of.reads	—	Specifies bases to ignore when doing rescue.
wt.rescue.max.mismatches.allowed	8 for F3 reads, and 6 for F5 reads	The maximum number of mismatches allowed in a rescued alignment.
insert.start	30	The lower limit of insert size for use in pairing calculations.
insert.end	100000	The upper limit of insert size for use in pairing calculations.
f3.read.length	50	The length of the F3 read.
f5.read.length	25	The length of the F5 read.

Table 12 Paired-end WTA input and output parameters

Parameter	Default	Description
Input parameters		
genome.reference	—	The path to the file that contains the reference genome.
filter.reference	—	The path to the file that contains the filter reference.
gtf.file	—	The path to the file containing the .gtf file.

Table 12 Paired-end WTA input and output parameters

Parameter	Default	Description
f3.reads.file	—	The path to the .*.csfasta file containing the F3 color reads.
f3.qual.file	—	The path to the *.qual file for the F3 reads.
f5.reads.file	—	The path to the *.csfasta file containing the F5 color reads.
f5.qual.file	—	The path to the *.qual file for the F5 reads.
Output parameters		
pairing.output.directory	output/paired_end/mapping	
pairing.output.bam.file	wt.pe.bam	The name of the *.bam file.

Tertiary analysis

In the context of BioScope™ Software, tertiary analysis refers to data analysis that takes place after reads are mapped. In the BioScope™ Software WTA single-read and paired-end pipelines, tertiary analysis is composed of counting exons, calculating coverage, and finding junctions. The *.bam file produced in the mapping portion of the pipeline is the principal input for each tertiary analysis tool:

- CountTags
- Sam2Wig
- JunctionFinder
- SASR_JunctionFinder

Count exons with the CountTags tool

Use the CountTags tool to count the number of reads that align within genomic features. The CountTag tool takes the following as input:

- A set of read alignments (*.bam file)
- A set of stringency parameters
- A set of genome annotations (*.gtf file)

CountTags tool algorithm description

Stringency parameters govern the set of alignments in a *.bam file. These alignments are considered in counting. Reads that do not meet these stringency parameters are filtered out and do not contribute to the result.

Three parameters govern stringency:

- Filter alignment mode
- Minimum score
- Minimum mapping quality

The filter alignment mode governs how multi-mapping reads are handled. In the single-read mapping pipeline, the option settings for filter alignment mode are all, unique, and primary (see Table 13). For the paired-end mapping pipeline, only *all* is a valid selection.

Table 13 Alignment filter nodes

Mode	Description	Pipeline
All	All alignments are considered.	Use in Paired-end and single-read.
Unique	Only unique alignments are considered. The score clear zone is used to determine if an alignment is unique.	Use in single-read pipeline.
Primary	Only primary alignments are considered.	Use in single-read pipeline.

If a minimum mapping quality is selected, only alignments with the minimum quality or greater are considered. Likewise, if a minimum score is selected, only alignments having the minimum score are considered.

Filtering by unique or top criteria is included mainly for legacy reasons. It is recommended that you use only default setting of mapping quality to filter reads.

Mapping quality incorporates an assessment of score and uniqueness. To filter using only mapping quality specify a minimum mapping quality, and set the alignment filter mode to all, and specify a minimum mapping quality. This works.

In previous releases of BioScope™ Software, filtering single-read alignments using the unique alignment filter mode was recommended. When unique filtering is enabled, only alignments that satisfy the following criteria pass the filter:

- The total number of alignments for the read is less than the maximum allowed to be reported in mapping (the Z parameter).
- The alignment must be the single best alignment of the set of alignments for the read.
- The score of the alignment must be sufficiently better than that of any suboptimal hits. If no suboptimal hits are present, a suboptimal hit is assumed to exist with a score one less than the minimum possible score for the anchor size and mismatch level used in mapping. The difference between the score of the alignment score and the best suboptimal alignment must be greater than or equal to the clear zone.

Each feature defined in the genome annotation is assigned a tag count. A tag count is the total number of alignments that pass the stringency filters and are consistent with the annotation. The criteria for counting an alignment toward a feature are:

- The alignment must overlap the feature's numeric coordinates (contig, start, end).
- The alignment strand orientation must be consistent with the feature strand orientation.

  Note: F3 reads align to the sense genomic strand; F5 reads align to the antisense genomic strand.
- If the alignment does not span an intron, the alignment must include no more than three bases outside the feature.
- If the alignment spans an intron, the position of the exon-intron boundary must match in the alignment and the feature.

RPKM (Reads Per Kilobase of exon model, per Million mapped reads), which is a normalized measure of expression, is also reported.

CountTag tool parameter description

Table 14 describes parameters that you can configure for the CountTags tool.

Table 14 Counttags tool algorithm, input, and output parameters

Parameter	Default	Description
wt.counttag.run	1	The value of 1 enables the CountTags tool in the pipeline.
Algorithm parameters		
wt.counttag.alignment.filter.mode	all	Specifies the alignment filter mode. You must select all, unique or primary.
wt.counttag.min.mapq	10	The minimum mapping quality for an alignment to be counted.
wt.counttag.score.clear.zone	5	The clear-zone specified when applying the unique alignment filter mode. The clear-zone specification does not affect other filter modes.
wt.counttag.min.alignment.score	10	The minimum score for an alignment to be counted.
Input parameters		
wt.counttag.exon.reference	Same as *.gtf file for alignment	The path to the *.gtf file used for counting.
wt.counttag.input.bam.file	*.bam output from alignment	The path to the *.bam file that contains the aligned reads.
Output parameters		
wt.output.dir	output/single_read/ counttag	The name of the directory in which results are saved.
wt.output.file.name	countagresult.txt	The name of the results file.

CountTags tool output format

The output format of the CountTags tool follows the same *gtf file format as the supplied genome annotation. The score field of the *.gtf file contains the tag count. RPKM is reported in the attributes field.

Note: You can invoke the CountTags tool as a standalone shell script from the bin directory in the `bioscope` bin directory.

Compute coverage with the Sam2Wig tool

The purpose of the Sam2Wig tool is to produce a *.wig-format coverage file that contains the calculated coverage for each genome strand.

Sam2Wig tool workflow description

Coverage in this context is an integer quantity calculated at every genomic position indicating the number of alignments covering the position. The strand orientation of reads is relevant when analyzing data generated using an RNA-Seq kit. The kit produces reads that preserve the strandedness of the molecule being sequenced: F3 reads align to the sense strand, and F5 reads align to the antisense strand. The Sam2Wig tool interprets alignment orientation in the context of the SOLiD read type (F3 or F5) and calculates coverage for each strand.

The *.bam file is the sole input for calculating coverage. Alignment stringency filtering is handled in a manner identical to that for CountTags.

Coverage results are reported in *.wig format and the results can be visualized with genome browsers such as the Broad Institute's IGV or the UCSC Genome Browser.

Sam2Wig tool parameters

The table describes the algorithm, input and output parameters for the Sam2Wig tool.

Table 15 Sam2Wig tool algorithm, input, and output parameters

Parameter	Default	Description
wt.sam2wig.run	1	A value of 1 enables the same2wig tool in the pipeline.
Algorithm parameters		
wt.sam2wig.alignment.filter.mode	all	Specifies the alignment filter mode. You must select from all, unique, or primary.
wt.sam2wig.min.mapq	10	The minimum mapping quality for an alignment to be counted.
wt.sam2wig.score.clear.zone	5	The clear zone that is specified when applying the unique alignment filter mode. This parameter does not affect on other filter modes.
wt.sam2wig.min.alignment.score	0	The minimum score required for an alignment to be included in coverage calculations.
Input parameters		
wt.sam2wig.input.bam.file	*.bam output from alignment	The path to the *.bam file containing the aligned reads.
Output parameters		
wt.sam2wig.output.dir	—	The name of the results directory.
wt.sam2wig.basefilename	coverage	The name of the results file.

Find junctions with the JunctionFinder tool

The JunctionFinder tool detects junctions between adjacent transcribed exons.

The tool takes as input:

- A *.bam file of paired-end reads sequenced from transcribed RNA
- A *.gtf file
- A *.fasta file of the reference genome

The tool produces three junction files:

- All detected junctions
- Junctions interpreted as arising from alternate gene splicing
- Junctions interpreted as arising from inter-gene fusion

JunctionFinder algorithm overview

The JunctionFinder tool consists of three major algorithms:

- Single-read JunctionFinder
- Paired-end JunctionFinder
- Evidence evaluator

The algorithms treat the input reads as two kinds of complementary evidence:

- A read is considered a piece of single-read evidence for a junction X-Y between two exons if its sequence overlaps both the 3′ end of exon X, and the 5′ end of exon Y, and its disjoint overlapping sequences add up to the read itself (see Figure 10).
- A paired-end fragment is considered paired-end evidence for a junction X-Y if one read of the pair maps uniquely to exon X and the other maps uniquely to exon Y (see Figure 10).

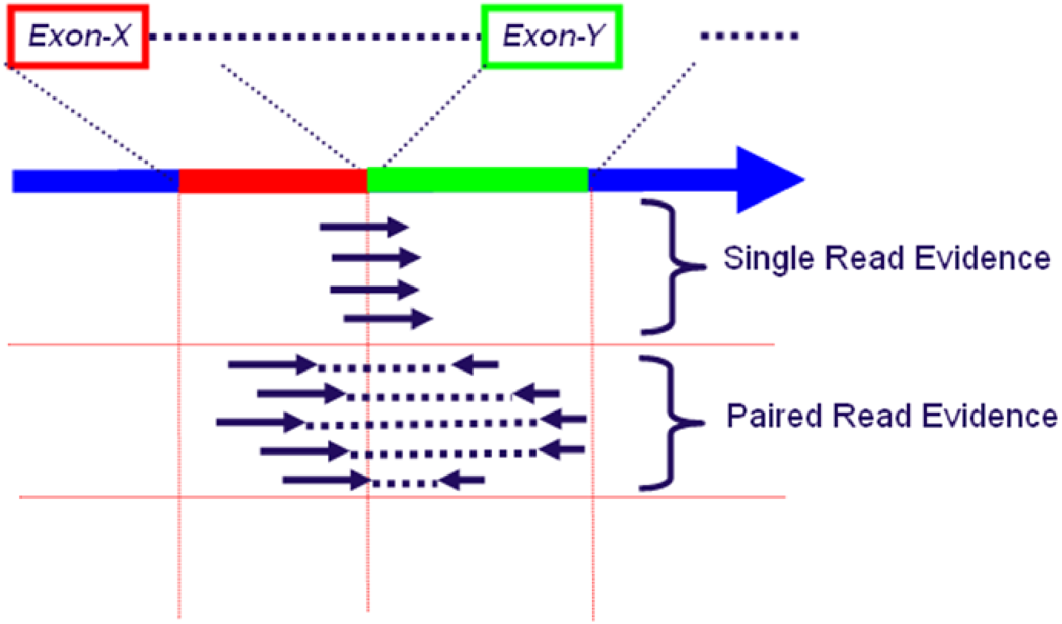


Figure 10 Single-read and paired-end evidence

In Figure 10, the upper two rectangles depict two exons X and Y that have been transcribed to a transcript, as shown by the multicolored arrow in Figure 10. The short arrows depict reads that form evidence. Single-read evidence has sequences that overlap the exon sequences; the reads are inferred to *span* the transcribed junctions. Paired-end evidence has reads that map to the genomic exons. They are interpreted to *bridge* the transcribed junction, depending on their insert size.

The first two algorithms, single-read JunctionFinder and paired-end JunctionFinder, store their candidate junctions, each with a count of evidence, in a candidate data structure. The Single-Read JunctionFinder considers only single-read evidence. The Paired-End JunctionFinder considers only paired-end evidence.

The evidence evaluator, which is the third algorithm, gathers the candidates from single-read JunctionFinder and the paired-end JunctionFinder, and combines both types of evidence to call junctions. The overall workflow of the JunctionFinder module is shown in Figure 11.

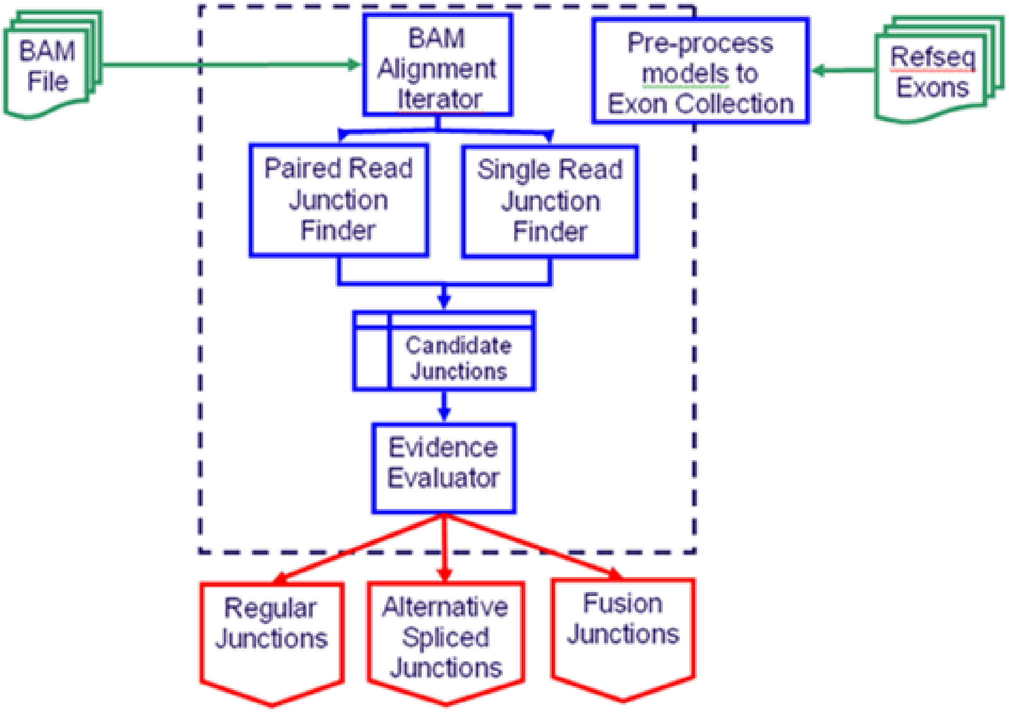

BAM File
BAM Alignment Iterator
Pre-process models to Exon Collection
Refseq Exons
Paired Read Junction Finder
Single Read Junction Finder
Candidate Junctions
Evidence Evaluator
Regular Junctions
Alternative Spliced Junctions
Fusion Junctions

Figure 11 JunctionFinder tool workflow

JunctionFinder tool algorithm description

The input *.gtf file contains the transcript annotations, which describe the intron-exon boundaries, and coding and non-coding elements of the genes. The JunctionFinder tool also takes as input a *.fasta file that contains the reference genome. The tool uses the reference genome to create an internal exon *.fasta file by extracting the sequence for each exon in the *.gtf file. The sequence is used by the single-read JunctionFinder, but not by paired-end JunctionFinder.

In the first part of the algorithm, JunctionFinder process the *.gtf file and the reference file to build the data structures necessary to efficiently compute junction evidence. Preparing the human genome for processing using the exon *.fasta file that was generated in the mapping stage typically takes less than one minute to complete.

In the second part of the algorithm, the input *.bam file contains pre-mapped alignments for the reads. The single-read and paired-end JunctionFinders examine every alignment individually to determine whether the alignments provide evidence for junctions. To ensure that the single-read and paired-end JunctionFinders count reads, not alignments, only records that have already been designated primary alignments are processed. Therefore, there is only one record per read.

Paired-End JunctionFinder description

The paired-end JunctionFinder considers reads where both reads of a pair are uniquely mapped. For each read, the paired-end JunctionFinder uses a function that quickly finds the exon that corresponds to the genomic alignment of the read. The algorithm uses the function to calculate if the two reads of a pair fall on different exons. If the two reads do fall on different exons, the algorithm counts the pair as evidence of a junction from the source exon, which is the one that mapped to the F3 read, to the destination exon, which is the one that mapped to the F5 read. This junction can be inferred because SOLiD^{TM} RNA libraries are strand-specific.

When the paired-end JunctionFinder deposits a candidate and its evidence into the data structure of the candidate, it also stores the start position of the mate alignment, to allow the Evidence Evaluator to determine the unique evidence count.

Single-Read JunctionFinder

For partially-mapped or even unmapped reads, the single-read JunctionFinder uses a suffix array-based algorithm (SASR) to efficiently compute the candidate junctions for which the read provides evidence.

When the single-read JunctionFinder deposits a candidate and its evidence into the candidates data structure of the candidate, it also stores the length of the overlap with the left exon. This value allows the Evidence Evaluator to determine the unique evidence count.

The core of the single-read algorithm has three steps. For each read in the *.bam file, the algorithm does the following:

1. Finds all exons that overlap the read on the left (see Figure 12).

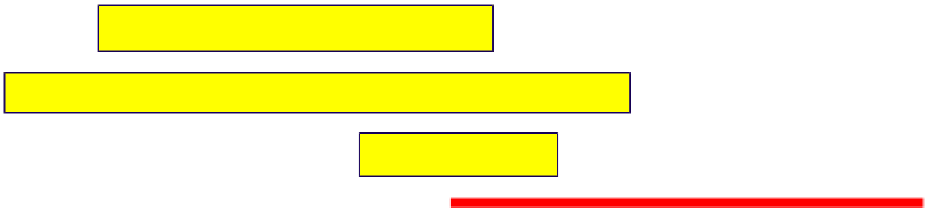


Figure 12 Exons that overlap the read on the left

2. Finds all exons that overlap the read on the right (see Figure 13).

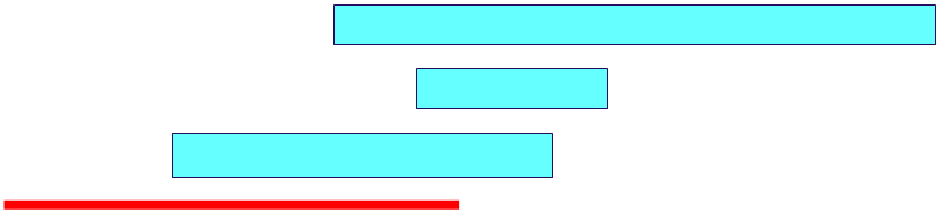


Figure 13 Exons that overlap the read on the right

3. Examines every pair of left-overlap/right-overlap exons (see Figure 14). If the pair meets requirements, the algorithm registers the read as evidence of a junction.

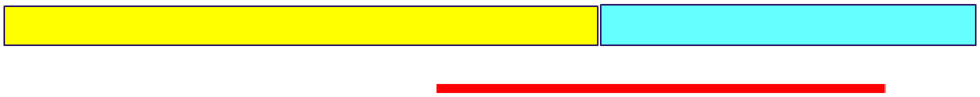
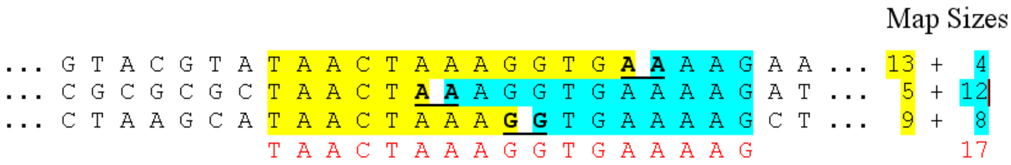


Figure 14 Left-overlap/right-overlap exons

To find the overlapping exons, the algorithm uses an efficient technique based on suffix arrays.

An exon-exon pair meets requirements if the exon overlaps satisfy certain length properties. In base space, a read provides evidence of a junction between an exon *e* and an exon *f* if and only if:

- Exon *e* maps to the prefix of the read.
- Exon *f* maps to the suffix of the read.
- The sum of the two map lengths is equal to the length of the read (see Figure 15).

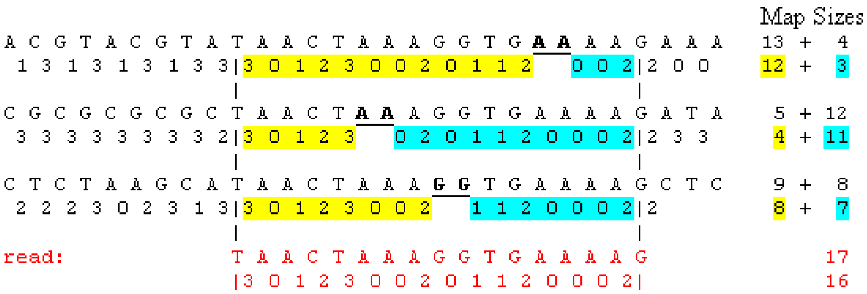


```
                                                                  Map Sizes
... G T A C G T A T A A C T A A A G G T G A A A A G A A ...  13 +  4
... C G C G C G C T A A C T A A A G G T G A A A A G A T ...   5 + 12
... C T A A G C A T A A C T A A A G G T G A A A A G C T ...   9 +  8
                  T A A C T A A A G G T G A A A A G                17
```

Figure 15 Three artificial exon pairs aligned with a read

Referring to Figure 15, the junction is marked in bold underline. The read serves as evidence for all three exon pairs aligned with a read. The lengths of the aligning suffix and prefix from each pair must add up to 17 bases, which is the length of the read.

The single-read JunctionFinder works directly in color-space. In color-space, conditions “a” and “b” remain unchanged. You must modify condition “c”. The color-space sketch corresponds to the base space sketch in Figure 16. The color-space sketch shows that two fused exons introduce an additional color between them. This additional color is not internal to either exon in the pair.



```
                                                                Map Sizes
A C G T A C G T A T A A C T A A A G G T G A A A A G A A A       13 +  4
 1 3 1 3 1 3 1 3 3|3 0 1 2 3 0 0 2 0 1 1 2     0 0 2|2 0 0      12 +  3
                  |                                 |
C G C G C G C G C T A A C T A A A G G T G A A A A G A T A        5 + 12
 3 3 3 3 3 3 3 3 2|3 0 1 2 3     0 2 0 1 1 2 0 0 0 2|2 3 3       4 + 11
                  |                                 |
C T C T A A G C A T A A C T A A A G G T G A A A A G C T C        9 +  8
 2 2 2 3 0 2 3 1 3|3 0 1 2 3 0 0 2     1 1 2 0 0 0 2|2           8 +  7
                  |                                 |
read:             T A A C T A A A G G T G A A A A G                 17
                  |3 0 1 2 3 0 0 2 0 1 1 2 0 0 0 2|                 16
```

Figure 16 Color alignment

In Figure 16, the two exons that are spliced together introduce an additional color between them. The color is present in the read, but not in the aligned exons.

Therefore, a pair of exons *e* and *f* meets requirements for evidence if:

- Exon *e* maps to the prefix of the read.
- Exon *f* maps to the suffix of the read, and
- The sum of the two map lengths (in colors) plus 1 is equal to the length of the read (in colors).

When the conditions in the bullet list (above) are in place, the algorithm can be described more formally.

SASR_JunctionFinder

Input files

The SASR_JunctionFinder tool uses the following files as input:

- A list of exons. Each exon includes its color sequence and the reverse of that sequence.
- A list of color-space reads (the *.bam file).

Output files

The output of the SASR_JunctionFinder tool is a list of exon-exon junctions. Each entry in the list includes the number of reads that provide evidence.

Run the SASR_JunctionFinder tool

SASR_JunctionFinder tool method

```
1. Read in and process the exon list.
2. For each admissible read:
   a. Let $L$ be the set of exons that map to the prefix of the read.
   b. Let $R$ be the set of exons that map to the suffix of the read.
   c. For each exon $e$ in $L$, and each exon $f$ in $R$,
         if $overlap(e, r) + overlap(f,r) + 1 = length(r)$, then
            register read $r$ as evidence for junction $e$-$f$.
3. Output the list of junctions and evidence.
```

SASR_JunctionFinder tool evidence evaluator

After the single-read and paired-end JunctionFinders have fully processed the *.bam file, the evidence evaluator evaluates the existing evidence and makes junction calls based on user-defined thresholds.

The default thresholds require at least one single-read and one paired-end evidence to call a junction. Special junctions, such as alternative splicing and fusion, are called, as described in the following three paragraphs.

The Evidence Evaluator also counts the number of unique pieces of evidence. In the case of paired-end evidence, this is the number of unique mate-alignment start positions, and in the case of single-read evidence, it is the number of unique overlap lengths for the "left" exon.

The candidate's data structure stores the evidence for junctions as properties of its exons. Each exon points to other exons for which there is evidence of a junction, where this exon is a source, and the pointed-to exon is the destination. The data structure of the candidate corresponds to a directed exon evidence graph.

In a junction evidence graph, nodes are exons, and each directed edge corresponds to junction evidence from a source exon to a destination exon. Each edge has two values:

- Unique SASR evidence
- Unique PR evidence

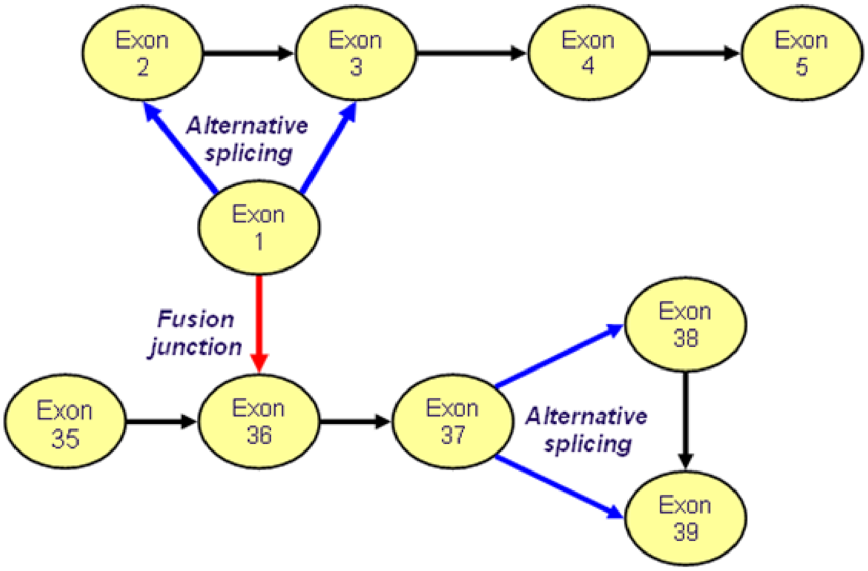


Figure 17 A directed exon evidence graph

Referring to Figure 17, the black arrows correspond to normal junctions, blue arrows correspond to alternative splice junctions, and red arrows correspond to fusions.

Most junctions found are regular junctions, meaning junctions between exons of the same gene in the RefSeq expected order. However, more interesting special junctions are also detected during this evaluation step. Alternatively, spliced junctions are defined as multiple junctions from a given exon to two other exons of the same gene within the given sample. Fusion junctions are defined as junctions between exons of different genes.

Calling junctions and fusions with single read only

BioScope™ Software v1.2 includes a new configuration (ini) file that allows it to call junctions and fusions with fragment (F3 only) datasets. However, BioScope™ Software's fusion caller is designed to work with paired-end reads. Calling fusions with only fragment reads will likely result in a large number of false positives. We observed that, on the same dataset, BioScope™ Software calls 10 times or more fusions using only single-read evidence compared with calls using both paired-end and single-read evidence. We suggest rigorous post-filtering and sorting by total unique single-read evidence for validating those fusion calls.

For detecting known, same-gene junctions we observed that using only single-reads calls 80% of the junctions called by paired-end reads (see Figure 18). However, the remaining 20% usually contains lowly expressed or harder to detect junctions, and we observed a need to double the total number of reads in order to achieve the same sensitivity. Thus, detection of exon junctions with fragment reads is not recommended due to lower specificity and sensitivity.

Figure 18 compares single-read (SR) vs. paired-end (SR+PE) junction calling sensitivity. The left two bars show SR and SR+PE number of junction calls when using UHR barcode 1. The middle two bars are with UHR barcode 2. The right two bars show the increase in number of calls when the two barcodes are merged and calls are repeated with effectively double the amount of reads from the same library. For both SR and SR+PE, two unique evidences were required to call a junction. For SR+PE, at least one SR and at least one PE evidence was required as well.

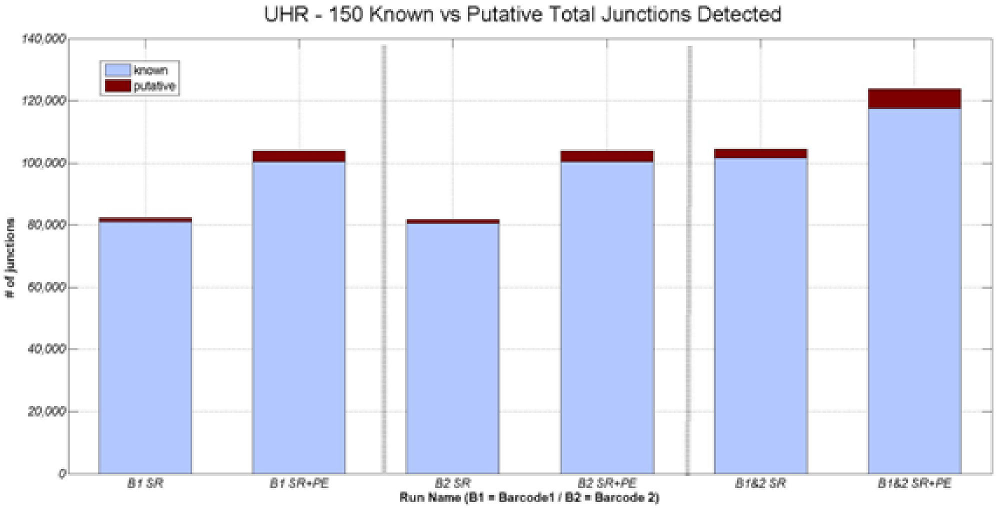


Figure 18 Comparison of single-read (SR) vs. paired-end (SR+PE) junction calling sensitivity with two barcodes

JunctionFinder parameters

Table 16 BioScope™ Software - JunctionFinder parameter description

Parameter name	Default value	Description
General options		
wt.junction.finder.input.bam	null	Single filename. Default calculated with *.bam analysis. Required.
wt.junction.finder.gtf.file	null	The *.gtf file used to create the gene model.
wt.junction.finder.input.exon .reference	null	Single file name. Default calculated with pipeline. Required. default value: null.
wt.junction.finder.output.dir	null	The junction, alt splicing, and fusion outputs are all written to this directory. Required. Default value: null.
wt.junction.finder.min.exon.length	25	All exons shorter than this are removed: • Minimum value: 0 • Maximum value: . (Not required).
wt.junction.finder.first.read.max.read.length	50	Auto populate if possible: • Minimum value: 1 • Maximum value: 50 (Required)
wt.junction.finder.second.read.max.read.length	25	Auto populate if possible: • Minimum value: 1 • Maximum value: 25 (Required).
Single read junction evidence collection		
wt.junction.finder.single.read	1	1=run single read fusion finding: • Minimum value: 0 • Maximum value: 1 (Not required).
wt.junction.finder.single.read.min.mapq	0	A read that has mapping quality lower than this is inadmissible as evidence. • minimum value: 0 • maximum value: 255 (Not required).
wt.junction.finder.single.read.min.overlap	10	The minimum number of contiguous color positions that must align in a read-exon map. • Minimum value: 0 • Maximum value: 50 (Not required).

Table 16 BioScope™ Software - JunctionFinder parameter description *(continued)*

Parameter name	Default value	**Description**
wt.junction.finder.single.read.max.mismatches	2	The maximum number of mismatches allowed in a read-exon map. • Minimum value: 0 • Maximum value: 25 (Not required).
wt.junction.finder.single.read.clip.size	2	Progressive unit size for clipping at the end of read. • Minimum value: 0 • Maximum value: 25 (Not required).
wt.junction.finder.single.read.clip.total	10	Total size for clipping at the end of read • Minimum value: 0 • Maximum value: 25 (Not required).
wt.junction.finder.single.read.ReportMultihit	0	Report 0=none. 1=all. 2=first minimum value: 0 maximum value: 2 (Not required.)
wt.junction.finder.single.read.remap	0	0=false, 1=true. If true, SASR remaps reads that have already been mapped to a junction, and registers evidence for that remap. If false, SASR registers the evidence, but does not remap: • Minimum value: 0 • Maximum value: 1 (Not required).
wt.junction.finder.single.read.clip.5.prime	1	If true, (1), SASR clips both the 5' and the 3' end of the read. If false (0), SASR clips only the 3' end. • Minimum value: 0 • Maximum value: 1 (Not required.)
wt.junction.finder.single.read.min.read.length	37	SASR considers shorter reads (in number of colors) to be inadmissible as evidence. • Minimum value: 0 • Maximum value: 100 (Not required).
Paired read junction evidence collection		
wt.junction.finder.paired.read	1	1=run paired-end fusion finding • Minimum value: 0 • Maximum value: 1 (Not required).

Table 16 BioScope™ Software - JunctionFinder parameter description *(continued)*

Parameter name	Default value	**Description**
wt.junction.finder.paired.read.min.mapq	10	A read-pair that has mapping quality lower than this is inadmissible as evidence. • Minimum value: 0 • Maximum value: 255 (Not required).
wt.junction.finder.paired.read.avg.insert.size	120	Average insert size of the paired-end library. • Minimum value: 0 • Maximum value: 1.00E+08 (Not required).
wt.junction.finder.paired.read.std.insert.size	60	Standard deviation of the insert size for the paired-end library. • Minimum value: 0 • Maximum value: 1.00E+08
Combined caller for junction		

The parameters below allow JunctionFinder to combine evidence from single-reads and from paired-reads to call junctions, alternative splices, and fusions, respectively. In each case, they will report an event (for example, a fusion) if SRE>x and PRE>y and ((SRE+PRE)>z or MAX(SRE,PRE) > w), where
SRE = unique single-read evidence,
PRE = unique paired-read evidence, and
x, y, z, and w are specified by the parameters below.		

wt.junction.finder.single.read.min.evidence.for.junction	1	x in SRE>x • Minimum value: 0 • Maximum value: 1 (Not required).
wt.junction.finder.paired.read.min.evidence.for.junction	1	y in PRE>y • Minimum value: 0 • Maximum value: . (Not required).
wt.junction.finder.combined.min.evidence.for.junction	2	z in (SRE+PRE)>z • Minimum value: 0 • Maximum value: . (Not required).
Combined caller for (same gene) alternative splicing		
wt.junction.finder.single.read.min.evidence.for.alt.splice	1	x • Minimum value: 0 • Maximum value: . (Not required).

Table 16 BioScope™ Software - JunctionFinder parameter description *(continued)*

Parameter name	Default value	**Description**
wt.junction.finder.paired.read.min.evidence.for.alt.splice	1	y • Minimum value: 0 • Maximum value: . (Not required).
wt.junction.finder.combined.min.evidence.for.alt.splice	3	z • Minimum value: 0 • Maximum value: . (Not required).
Combined caller for gene fusion		
wt.junction.finder.single.read.min.evidence.for.fusion	2	x • Minimum value: 0 • Maximum value: . (Not required.)
wt.junction.finder.paired.read.min.evidence.for.fusion	2	y • Minimum value: 0 • Maximum value: . (Not required).
wt.junction.finder.combined.evidence.for.fusion	4	z Minimum value: 0 Maximum value: . (Not required.)
Other options		
wt.junction.finder.show.same.exon.pairs	0	Set 1 for debug mode or interest in pairs within same exon Minimum value: 0 Maximum value: 1 (Not required).
wt.junction.finder.output.format	1	Output format parameter. Allowed values: • 1 = tabular output format • 2 = BED output format • 3 = all formats With option 3, both tabular, BED, and also additional "seq" files are created separately. Seq files contain 50 base pairs of sequences from each end of a junction exon and are useful for validation. Minimum value: 0 Maximum value: 3 (Not required).

Input files

The SASR_JunctionFinder tool takes the following files as input:

- *.gtf file
- *.fasta genome reference file
- *.bam file of genomic alignments with reads

Output files

Junction, splicing, and fusion output files

The JunctionFinder tool generates six output files (see Table 17). Log and stats output files include run information and tables that summarize the number of junctions detected using the program. Three output files are generated with the *.tab extension for regular junctions, alternatively-spliced junctions, and fusion junctions.

Table 17 Junction tab-delimited output format

Field name	Description
Exon-1	The gene id followed by the exon order on the gene.
Exon-1-reference	The reference *.fasta file.
Exon-1 strand	± strand.
Exon-1 start	The start position.
Exon-1 end	The end position.
Exon-2	The gene id followed by the exon order on the gene.
Exon-2-reference	The reference *.fasta file.
Exon-2 strand	± strand.
Exon-2 start	The start position.
Exon-2 end	The end position.
Exon-1-size	The size of the first exon.
Exon-2-size	The size of the second exon.
Exon-1-readcount	The number of reads that map on the first exon.
Exon-2-readcount	The number of reads that map on the second exon.
Exon-1-F3-RPKM	The Reads Per KB Per Million Reads (RPKM) exon-1 from F3.
Exon-2-F3-RPKM	The RPKM reads (RPKM) exon-2 from F3.
Exon-distance	Distance between two exons. The exon-distance is not applicable if it is on a different chromosome.
Total-PR-evidence	The total paired-end evidence for the junction.
Unique-PR-evidence	The unique paired-end evidence for the junction.
Total-SR-evidence	The total single-read evidence for the junction.
Unique-SR-evidence	The unique single-read evidence for this junction.
JCV	A Junction confidence value.
Known	"K" if a junction is known, and "P" if a junction is putative.
E1-all-genes	The list of all genes to which exon-1 was mapped.

Table 17 Junction tab-delimited output format *(continued)*

Field name	Description
E2-all-genes	The list of all genes to which exon-1 was mapped.
Other	Other information provided about the junction.

The columns that are most relevant to the JunctionFinder are "unique paired-end" and "unique single-read evidence". Also of interest are the two metrics RPKM and JCV.

The purpose of the RPKM (Reads Per Kilobase of exon sequence, per Million reads) metric is to provide a normalized exon expression level. This metric is calculated with the formula:

$$RPKM = 10^9 \times \frac{ExonReadCount}{TotalReadCount \times ExonLength}$$

The RPKM value is also reported by the CountTags plug-in (see "CountTags tool algorithm description" on page 61).

The purpose of the Junction Confidence Value (JCV) metric is to aid in the detection of false positives and in other decisions that depend on a confidence level. Depending on the coverage of the sequencing experiment and exons being tested, the algorithms might generate results that require different user-defined thresholds to detect most likely fusions. It is possible that the major contributor of false positives is highly-expressed genes for which there is a considerable random chance of encountering a fusion event due to sequencing errors and mapping to homologous exons. A statistical confidence metric (see below) was developed to detect such false positives and assign a confidence level to the evidenced junction.

$$JCV_{j_{x-y}} = \sum_{i=1}^{n} PQV_i - 10\log_{10}\ (EEM_{j_{x-y}})$$

Equation 1. Junction Confidence Value (JCV)

$$EEM_{j_{x-y}} = \frac{\frac{RC_x}{l_x}}{\mu_T + 3\times\sigma_T} \times \frac{\frac{RC_y}{l_y}}{\mu_T + 3\times\sigma_T}$$

Equation 2. Error expectation metric (EEM)

where PQV_i is the Phred-scale pairing quality value for the i'th unique paired read evidence for a candidate junction j_{x-y} and x and y are the junction exons. For each unique single read evidence, the PQV_i is set to 10. If there are multiple alignments *for* a given unique start point, take the *PQV* of the first such alignment. *RC* is the absolute proper mapped read count for the corresponding exon and is the length of the exon; μ_T and σ_T are the mean and standard deviation of the insert size for the current experiment, *T*.

Error expectation metric (EEM) is used to quantify highly expressed junctions. This metric is hard to calculate due to genome complexity and homology of exons. Our estimation considers the number of reads mapped to the exons, the length of exons and a conservative insert range.

After the equation is calculated, a JCV that is larger than 100 is set to 100 and if it is smaller then 0 it is set to 0. If a JCV approaches 100, it is more likely to be a real junction.

Examples

For a junction detected between exon-1 of size 5,000 and exon-2 of size 400, with mean insert 100 and standard deviation 33.3 bp, assume in case-1, there were three unique junction evidences with PQV 30, 20 and 40 respectively and in case-2 only a single unique evidence with PQV 20. There were 900 properly mapped alignments to exon-1 and 100 such alignments for exon-2. In case-3 has 3 unique evidences similar to case-1, but the exons are assumed to have 100 times more coverage each. The junction confidence value for a junction between these exons would be:

$$30+20+40-10\times\log_{10}\left(\frac{900}{5000}\times\frac{100}{400}\times 200^{2}\right)=90-10\times\log_{10}(1800)\approx 50\text{ to }60$$

Equation 3. Case-1.3 unique junction evidence between 9x and 12x exons

$$20-10\times\log_{10}\left(\frac{900}{5000}\times\frac{100}{400}\times 200^{2}\right)=20-10\times\log_{10}(1800)\approx 0$$

Equation 4. Case-2.1 unique junction evidence between 9x and 12x exons

$$30+20+40-10\times\log_{10}\left(\frac{90000}{5000}\times\frac{10000}{400}\times 200^{2}\right)=90-10\times\log_{10}(18{,}000{,}00)\approx 10\text{to}20$$

Equation 5. Case-3. 3 unique junction evidence between 900x and 1200x exons

A simplified table of outputs is shown in Table 18. Refer to the Bioscope™ Software demos or applications in the examples directory for full output and example alternative splicing/fusion output.

Refer to the Bioscope™ Software examples directory for full examples of output and an example of alternative splicing/fusion output.

Table 18 Simplified junction tab output

E1	E1-reference	s	E1-start/ E1-end	E2	E2-reference	s	E2-Start/ E2-End	Unique-PR	Unique-SR
AGRN-1	chr1	+	886872/ 886993	KLHL17-4	chr1	+	887069/ 887290	3	2
KLHL17-8	chr1	+	888580/ 888747	KLHL17-9	chr1	+	889163/ 889251	1	1
AGRN-11	chr1	+	969352/ 969500	AGRN-12	chr1	+	969577/ 969682	2	1
AGRN-14	chr1	+	970602/ 970766	AGRN-15	chr1	+	970976/ 971119	2	1

Table 18 Simplified junction tab output *(continued)*

E1	E1-reference	s	E1-start/ E1-end	E2	E2-reference	s	E2-Start/ E2-End	Unique-PR	Unique-SR
AGRN-17	chr1	+	971403/ 971508	AGRN-15	chr1	+	971640/ 971978	6	2
AGRN-20	chr1	+	972570/ 972697	AGRN-21	chr1	+	972816/ 972930	2	3
AGRN-21	chr1	+	972816/ 972930	AGRN-22	chr1	+	973019/ 973138	2	1
AGRN-26	chr1	+	974809/ 975038	AGRN-27	chr1	+	975146/ 975280	3	0
AGRN-27	chr1	+	975146/ 975280	AGRN-28	chr1	+	975476/ 975572	3	1
PUSL1-4	chr1	+	1234697/ 1234846	PUSL1-5	chr1	+	1234924/ 1235094	2	0
PUSL1-6	chr1	+	1235877/ 1235931	PUSL1-7	chr1	+	1236152/ 1236314	2	0
VWA1-0	chr1	+	1362170/ 1362727	VWA1-3	chr1	+	1364324/ 136600	4	0
MIB2-3	chr1	+	1548632/ 1548942	MIB2-4	chr1	+	1549017/ 1549188	5	1
MIB2-4	chr1	+	1549017/ 1549188	MIB2-5	chr1	+	1550038/ 1550144	1	1
MIB2-6	chr1	+	1550234/ 1550428	MIB2-7	chr1	+	1550529/ 1550671	2	1
MIB2-7	chr1	+	1550529/ 1550671	MIB2-8	chr1	+	1550789/ 1550896	2	1
MIB2-9	chr1	+	1551893/ 1551997	MIB2-10	chr1	+	1552080/ 1552242	2	2
MIB2-10	chr1	+	1552080/ 1552242	MIB2-11	chr1	+	1552317/ 1552450	3	0
MIB2-11	chr1	+	1552317/ 1552450	MIB2-12	chr1	+	1552539/ 1552687	2	0
MIB2-14	chr1	+	1553262/ 1553422	MIB2-15	chr1	+	1553516/ 1553642	3	1

Browser Extensible Display (BED) output

The BED format was developed to extend the UCSC Genome Browser with user-defined tracks. BED is used to visualize the splice and fusion junctions in the UCSC Genome browser and in the IGV browser (see Figure 19 on page 80). For general documentation about the BED format, including information about all of the BED fields, go to:

genome.ucsc.edu/FAQ/FAQformat.html

For information about visualization software, see "Visualizing *.bam output" on page 299.

Each line in the track defines a junction where chromStart is the smaller of the coordinates of and chromEnd is the greater.

There are two blocks because a junction typically contains two exons. BlockSizes are the lengths of the exons. The block starts the beginning of the exons. When fusions on different strands or chromosomes, two lines are added to the output, with each line representing one chromosome. Different colors are used to color-code different types:

Figure 19 on page 80 shows the Upstream Hypersensitive Region (UHR) gene region displayed with the Integrative Genomics Viewer (IGV) for positions 3,530,193 to 3,548,355 of Human Chr-1. The following sections describe the tracks in Figure 19.

WIG (x2) tracks

The top two tracks show the genomic coverage using the negative strand and positive strand generated by the Bam2Wig tool (Max: 100 coverage).

BAM track

The middle track shows the alignments from the *.bam file. For display purposes, reads are filtered with MAPQ threshold of 45 (a stringent filter). Bases with quality value 5 to 20 are shaded.

BED track

The fifth track shows the junctions detected by the Junction Finder (BED file). As shown in the figure, all junctions detected are "known" and so are shaded in green.

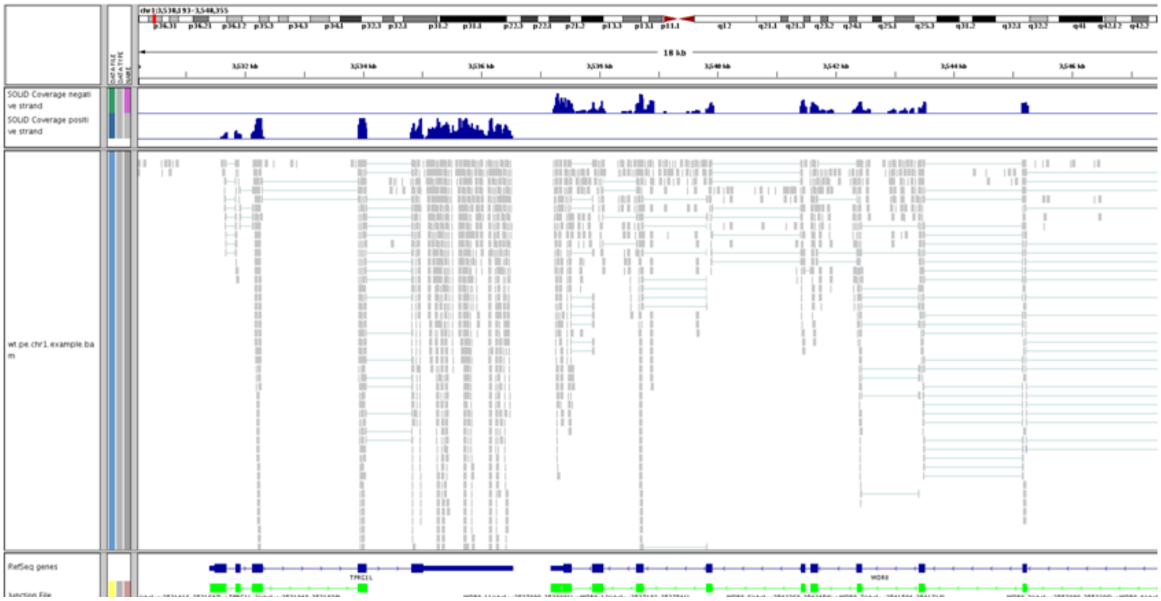


Figure 19 BED-IGV extended track example

Using *.gtf files in WTA pipelines

WTA pipelines use genome annotation files in *.gtf format. Go to the following URLs to see the *.gtf format explained in detail:

genes.cse.wustl.edu/GTF2.html

genome.ucsc.edu/FAQ/FAQformat.html#format4

The *.gtf file must match the genome reference to ensure that the WTA pipelines work correctly.

IMPORTANT! Make sure the *.gtf file is for the same genome assembly as the *.fasta file, and that matching sequence identifiers are used. Gene and transcript identifiers in the *.gtf file must be properly normalized. Identifier normalization is a known issue in *.gtf files from the UCSC Genome Browser, which is a popular source of *.gtf-formatted annotation. The UCSC *.gtf files always report the same value for gene and transcript IDs.

Bioscope™ Software includes scripts that transform a *.gtf file into the format required for use with WTA pipelines.

Formatting UCSC Genome Browser annotations for WTA pipelines

The UCSC Genome Browser has genome annotations available for many assemblies at

hgdownload.cse.ucsc.edu/goldenPath/

The *.gtf-formatted annotations available for download are not properly normalized by gene ID. The required content is present for each assembly in the file export of the refGene database table `database/refGene/txt/gz`.

For example, annotation for human genome build 18 is available at:

hgdownload.cse.ucsc.edu/goldenPath/hg18/database/refGene.txt.gz

Note: The *.gtf-formatted annotation is not in *.gtf format. You must convert the annotation before using it in WTA.

Convert the refGene.txt.gz file

Run the script `bin/refgene2gtf.sh` to convert the `refGene.txt.gz` file:

```
% gunzip refGene.txt.gz
```

```
% refgene2gtf.sh -i refGene.txt -o refGene.gtf
```

Genome annotations that are downloaded from the UCSC Genome Browser and converted by the annotation conversion script are optimal because they contain Human Genome Organization (HUGO)-style gene names. HUGO-style gene names allow interpretation when using a genome browser or reading reports.

The annotation conversion script works with the latest format of `refGene.txt` files. Assemblies, such as the rat genome, use an alternative format for the `refGene.txt` file. The `refgene2gtf.sh` script does not convert alternative formats.

Formatting ENSEMBL *.gtf files for WTA pipelines

The ENSEMBL website **ensembl.org**/ has *.gtf-formatted genome annotations available for many popular assemblies. Unlike the *.gtf files directly downloadable from UCSC (see "Using *.gtf files in WTA pipelines" on page 81), ENSEMBL *.gtf files are properly normalized by gene and transcript IDs.

http://www.ensembl.org/

ENSEMBL *.gtf files use gene accession numbers instead of HUGO-style gene names. ENSEMBL *.gtf files also use unprefixed sequence identifiers, such as 1,2,3….X,Y,MT. The ENSEMBL *.gtf files are incompatible with genome reference *.fasta files that have UCSC-style sequence IDs with the prefix "chr", for example, chr1, chr2, chr3….chrX, chrY, chrM.

Reformat the ENSEMBL .*gtf file

To reformat the ENSEMBL *.gtf file to use UCSC-style gene IDs, run `convert_ensembl_gtf.pl`:

```
% reformat_ensembl_gtf.pl Homo_sapiens.GR
```

WTA output file formats

The WTA paired-end pipeline produces *.bam files that are identical to those produced by the resequencing pipeline **samtools.sourceforge.net/SAM1.pdf**. However, the *.bam files from the WTA single-read pipeline differ from *.bam files produced elsewhere in BioScope™ Software. See Table 9 on page 54.

5 Run the Whole Transcriptome Data Mapping Tool

This chapter covers:

Map Whole Transcriptome introduction

Four parallel read mappings occur in WTA:

- Mapping to filter sequences
- Mapping to splice junction
- Mapping to the genome reference
- Exon mapping and pairing

Mapping jobs are divided and distributed across the available cluster resources, then mapping results are merged and sorted. For details about the mapping algorithm and the *.ini files associated with WT mapping and pairing, see "Whole Transcriptome Pipeline Concepts" on page 47.

GTF file format description

A *.gtf file is a more stringent version of a *.gff*. file. The first eight fields in the *.gtf file are the same as the first eight fields in a *.gff file. The group field in the *.gtf file has been expanded into a list of attributes. Each attribute consists of a type/value pair. Attributes must end in a semicolon and be separated from any following attribute by exactly one space.

A *.gtf file is provided to the `bioscope.sh` command via the wt.splext.genegtf file parameter. The *.gtf is the file that defines the genes and transcripts in the reference genomes. Details about the *.gtf format can be obtained at

mblab.wustl.edu/GTF2.html

A *.gtf file containing the human genes in the UCSC Genome browser's RefGene table is available from

www.solidsoftwaretools.com

IMPORTANT! The *.gtf files that are available directly from the UCSC Genome Browser are not appropriate for this tool because they do not group features by gene_id. BioScope™ Software requires this file to group exons from the same gene with common values in the gene_id field in the attributes column.

BioScope™ Software provides a program, `bin/refgene2gtf.sh,` that you can use to convert the RefGene table (RefGene.txt) from the UCSC Genome Browser to a *.gtf format appropriate for use with BioScope™ Software WTA tools. The RefGene.txt file is available from

hgdownload.cse.ucsc.edu/goldenPath/hg18/database/

BioScope™ Software has only tested this program with the UCSC human RefGene.txt. In cases where a RefGene.txt file is not available, a custom script for preparing the *.gtf file is required.

IMPORTANT! BioScope™ Software has only tested this program with the UCSC human RefGene.txt. In cases where a RefGene.txt file is not available, a custom script for preparing the *.gtf file is required

Run Whole Transcriptome Map Data

This section explains how to run Whole Transcriptome mapping from the command line or the web interface.

Complete the prerequisites

1. Complete the applicable prerequisites described in Chapter 3, "Before you Begin" on page 35.
2. Login to the BioScope™ Software cluster. Update the wt.ini file with information that applies to the experiment that you plan to run. See "Example wt.ini file" on page 86.
3. Convert RefGene.txt to RefGene.gtf:

   ```
   % refgene2gtf.sh =i refGene.txt -o refGene.gtf
   ```

Run WT Map Data from the command line

Although several different software programs are involved in the experiment, a single command generates all of the related programs required to complete the experiment. The *.plan file that is specified in the command syntax controls the order in which BioScope™ Software runs the related programs.

1. Connect to the BioScope™ Software cluster and login with a user ID that has write privileges on all of the directories that BioScope™ Software uses when the tool runs.

   If your user ID does not have write privileges on those directories, enter `su` at a command prompt and change to `bioscope` or another ID that has the correct privileges.
2. At a command prompt, enter:

   ```
   bioscope.sh -l filename.log filename.plan
   ```
3. Navigate to the /output/log directory. Open the log folder to check the status of your run.

Run Map WT Data from the web interface

1. Launch a browser and enter the BioScope™ Software URL.
2. Click **WT Map Data**.
3. Select the Genome Reference File (*.fasta).
4. Select the Filter Reference File (*.fasta).
5. Select the Gene *.gtf file.
6. Select F3 Reads File(*.csfasta).
7. Select F3 Quality Value file (*qual).
8. Select the Maximum Hits.
9. Select **Start WT Single Read**.

View the log folder to check when your experiment is complete.

Access the results files

For information about the *.bam file generated by a Whole Transcriptome experiment, see Appendix A, “File Format Descriptions” on page 295.

Example wt.ini file

The following section shows an example of the wt.ini file.

```
#
# DIRECTIONS: run = 1 to run the plugin, run = 0 to disable the
plugin.

###############################################################
#      Global settings for the pipeline run
###############################################################

import wt.sr.common.ini

###############################################################
#      Local settings for the pipeline run
###############################################################

#--------------------------------------------------------------
#                      Splice Junction Extractor
# Optional plugin.  Must be run before junction mapping.
# Parameters:wt.splext.genegtf.file, a genome reference file.
#  wt.splext.reference.file, a gene gtf file that matches the
genome reference file.
# Output:A junction reference file in ${intermediate.dir}/
spljunctionextraction.
# Description:Uses the gtf file to extract the splice junctions
from the genome reference.
#--------------------------------------------------------------

wt.spljunctionextractor.run = 1
wt.splext.genegtf.file = ${exons.gtf.file}
wt.splext.reference.file = ${reference.file}

#--------------------------------------------------------------
#                      Junction Mapping Plugin
# Optional plugin.  Splice junction extractor must be run before
junction mapping.
# Output:Resulting .ma file in ${intermediate.dir}/s_mapping/
junction_map.
# Description:Maps  the reads against the splice junction
reference.
#--------------------------------------------------------------

wt.junction.mapping.run = 1
```

```
#-------------------------------------------------------------
#                      Filter Mapping Plugin
# Optional plugin.
# Parameters:wt.filter.mapping.reference, a filter reference
file.
# Output:Resulting .ma file in ${intermediate.dir}/s_mapping/
filter_map.
# Description:Maps the reads against the filter reference.
#-------------------------------------------------------------

wt.filter.mapping.run = 1
wt.filter.mapping.reference = ${filter.reference.file}


#-------------------------------------------------------------
#                      Genomic Mapping Plugin
# Required plugin.
# Parameters:wt.genomic.mapping.reference,a genome reference
file.
# Output:Resulting .ma file in ${intermediate.dir}/s_mapping/
genomic_map.
# Description:Maps the reads against the goenome reference.
#-------------------------------------------------------------

wt.genomic.mapping.run = 1
wt.genomic.mapping.reference = ${reference.file}


#-------------------------------------------------------------
#                       Merge Plugin
# Required plugin.
# Parameters:wt.merge.reference.file, a genome reference file.
#   wt.merge.filter.reference.file, a filter reference file.
#   wt.merge.junction.reference.file, a junction reference file
from splice junction extraction.
#   wt.merge.qual.file, a color quality file.
#   wt.merge.tmpdir, temporary directory.
#   wt.merge.known.juntion.penalty, known junction penalty.
#   wt.merge.putative.junction.penalty, putative junction
penalty.
#   wt.merge.score.clear.zone, score clear zone.
#   wt.merge.min.junction.overhang, minimum junction overhang.
#   wt.merge.num.alignments.to.store, number of alignments to
store.
# Output:a .bam file in ${output.dir}/mapping.
# Description:Merge mapping results to produce a .bam file.
#--------------------------------------------------------------

wt.merge.run = 1

wt.merge.reference.file = ${reference.file}
wt.merge.filter.reference.file = ${filter.reference.file}
wt.merge.junction.reference.file = ${junction.reference.file}
wt.merge.qual.file = ${qual.file}
wt.merge.tmpdir = ${tmp.dir}
wt.merge.output.dir = ${merge.output.directory}
wt.merge.output.bam.file = ${merge.output.bam.file}

#wt.merge.known.juntion.penalty = 0
#wt.merge.putative.junction.penalty = 1
#wt.merge.score.clear.zone = 5
#wt.merge.min.junction.overhang = 8
#wt.merge.num.alignments.to.store = 1


#-------------------------------------------------------------
#                       Sam2wig Plugin
# Optional plugin.
# Parameters:wt.sam2wig.input.bam.file, a .bam file
#   wt.sam2wig.output.dir, output directory.
#   wt.sam2wig.basefilename, base file name.
#   wt.sam2wig.alignment.score, minimum alignment score.
#   wt.sam2wig.min.coverage, minimum coverage.
#   wt.sam2wig.wigperchromosome, wig per chromosome.
#   wt.sam2wig.alignment.filter.mode, filter mode.
#   wt.sam2wig.score.clear.zone, clear zone.
#   wt.sam2wig.min.mapq, minimum mapping quality.
# Output:a .wig file in ${output.dir}/sam2wig.
# Description:Generates a coverage .wig file from a .bam file.
#-------------------------------------------------------------

wt.sam2wig.run = 1

wt.sam2wig.input.bam.file = ${merge.output.directory}/
${merge.output.bam.file}
wt.sam2wig.output.dir = ${output.dir}/sam2wig
wt.sam2wig.basefilename = coverage

#wt.sam2wig.alignment.score = 0
#wt.sam2wig.min.coverage = 10
#wt.sam2wig.wigperchromosome = true
#wt.sam2wig.alignment.filter.mode = primary
#wt.sam2wig.score.clear.zone = 5
#wt.sam2wig.min.mapq = 10


#-------------------------------------------------------------
#                         Count Tag Plugin
# Optional plugin.
# Parameters:wt.counttag.input.bam.file, a .bam file
#   wt.counttag.exon.reference, an exons .gtf file.
#   wt.counttag.output.dir, output directory.
#   wt.counttag.output.file.name, output file name.
#   wt.counttag.score.clear.zone, clear zone.
#   wt.counttag.alignment.filter.mode, alignment filter mode.
#   wt.counttag.min.alignment.score, minimum alignment score.
#   wt.counttag.min.mapq, minimum map quality.
# Output:A counttag result file in ${output.dir}/counttag.
# Description:Counts tags from a .bam file.
#-------------------------------------------------------------
wt.counttag.run = 1

wt.counttag.exon.reference = ${exons.gtf.file}
wt.counttag.input.bam.file = ${merge.output.directory}/
${merge.output.bam.file}

wt.counttag.output.dir = ${output.dir}/counttag
wt.counttag.output.file.name = countagresult.txt

#wt.counttag.score.clear.zone = 5
#wt.counttag.alignment.filter.mode = primary
#wt.counttag.min.alignment.score = 0
#wt.counttag.min.mapq = 10


#-------------------------------------------------------------
#                       Junction Finder Plugins
# NOTE: The fusion caller is designed to work with paired-end 
reads.
# Calling fusions with fragment reads will likely result in a 
large number of false
# positives. Detection of exon junctions with fragment reads 
will have lower specificity
# and slightly lower sensitivity than exon junction detection 
with paired-end reads.
#
# Required plugin.
# Parameters:wt.junction.finder.input.bam, a .bam file.
#   exons.gtf.file, an exon .gtf file.
#   wt.junction.finder.input.exon.reference, an exon reference.
#   wt.genome.reference, a genome reference.
#   wt.junction.finder.min.exon.length, minimum exon length.
#   wt.junction.finder.first.read.max.read.length, first read 
length.
#   wt.junction.finder.first.read.max.read.length, Second read 
length.
#   wt.junction.finder.single.read, run single read fusion 
finding.
#   wt.junction.finder.single.read.min.mapq, minimum mapping 
quality for single reads.
#   wt.junction.finder.single.read.min.overlap, single read 
minimum overlap.
#   wt.junction.finder.single.read.max.mismatches, single read 
maximum mismatches.
#   wt.junction.finder.single.read.clip.size, single read clip 
size at the end of the read.
#   wt.junction.finder.single.read.clip.total, single read total 
size for clipping.
#   wt.junction.finder.single.read.ReportMultihit, single read 
report multiple hits.
#   wt.junction.finder.single.read.remap, single read remap.
#   wt.junction.finder.single.read.clip.5.prime, single read
clip the end of the 3' and 5' end of the read.
#   wt.junction.finder.single.read.min.read.length, single read
minimum read length considered.
#   wt.junction.finder.paired.read, paired read run.
#   wt.junction.finder.paired.read.min.mapq, paired read minimum
map quality.
#   wt.junction.finder.paired.read.avg.insert.size, paired read
average insert size.
#   wt.junction.finder.paired.read.std.insert.size, paired read
standard insert size.
#   wt.junction.finder.single.read.min.evidence.for.junctionm
single read minimum evidence for junction.
#   wt.junction.finder.paired.read.min.evidence.for.junction,
paired read minimum evidence for junction.
#   wt.junction.finder.combined.min.evidence.for.junction,
combined minimum evidence for junction.
#   wt.junction.finder.single.read.min.evidence.for.alt.splice,
single read minimum evidence for alternative splices.
#   wt.junction.finder.paired.read.min.evidence.for.alt.splice,
paired read minimum evidence for alternative splices.
#   wt.junction.finder.combined.min.evidence.for.alt.splice,
combined minimum evidence for alternative splices.
#   wt.junction.finder.single.read.min.evidence.for.fusion,
single read minimum evidence for fusions.
#   wt.junction.finder.paired.read.min.evidence.for.fusion,
paired read minimum evidence for fusions.
#   wt.junction.finder.combined.evidence.for.fusion, combined
minimum evidence for fusions.
#   wt.junction.finder.show.same.exon.pairs, show same exon
pairs.
#   wt.junction.finder.output.format, output format.
# Output:Files containing junctions, fusions, and alternative
splices
#   in ${output.dir}/junction_finder.
# Description:Finds junctions, fusions, and alternative splices.
#-------------------------------------------------------------

wt.exon.sequence.extractor.run = 1
wt.junction.finder.run = 1

wt.genome.reference = ${reference.file}
wt.gtf.file = ${exons.gtf.file}
wt.f5.exseqext.output.reference = ${intermediate.dir}/
exonsequenceextraction/exons_reference.fasta

wt.junction.finder.gtf.file = ${exons.gtf.file}
wt.junction.finder.input.exon.reference =
${wt.f5.exseqext.output.reference}
wt.junction.finder.input.bam = ${merge.output.directory}/
${merge.output.bam.file}
wt.junction.finder.output.dir = ${output.dir}/junction_finder

#wt.junction.finder.min.exon.length = 25
```

```
#wt.junction.finder.first.read.max.read.length = 50
#wt.junction.finder.second.read.max.read.length = 25

#wt.junction.finder.single.read = 1
#wt.junction.finder.single.read.min.mapq = 0
#wt.junction.finder.single.read.min.overlap = 10
#wt.junction.finder.single.read.max.mismatches = 2
#wt.junction.finder.single.read.clip.size = 2
#wt.junction.finder.single.read.clip.total = 10
#wt.junction.finder.single.read.ReportMultihit = 0
#wt.junction.finder.single.read.remap = 0
#wt.junction.finder.single.read.clip.5.prime = 1
#wt.junction.finder.single.read.min.read.length = 37

#wt.junction.finder.paired.read = 0
#wt.junction.finder.paired.read.min.mapq = 10
#wt.junction.finder.paired.read.avg.insert.size = 120
#wt.junction.finder.paired.read.std.insert.size = 60

#wt.junction.finder.single.read.min.evidence.for.junction = 2
#wt.junction.finder.paired.read.min.evidence.for.junction = 0
#wt.junction.finder.combined.min.evidence.for.junction = 2

#wt.junction.finder.single.read.min.evidence.for.alt.splice = 2
#wt.junction.finder.paired.read.min.evidence.for.alt.splice = 0
#wt.junction.finder.combined.min.evidence.for.alt.splice = 2

#wt.junction.finder.single.read.min.evidence.for.fusion = 2
#wt.junction.finder.paired.read.min.evidence.for.fusion = 0
#wt.junction.finder.combined.evidence.for.fusion = 2

#wt.junction.finder.show.same.exon.pairs = 0
#wt.junction.finder.output.format = 3
```

6 Run the Count Known Exons Tool

This chapter covers:

Count Known Exons introduction

This pipeline generates tag counts for annotated regions. Run this tool to extract reads that fall between a particular parameter.

IMPORTANT! Alignments must come from the same strand as the feature to contribute to the count. A non-gapped tag contributes to a feature's count if it overlaps the feature and has no more than three bases outside the feature. A gapped tag contributes to a feature.s count if one of its match regions terminates at a feature boundary

GTF file format description

A *.gtf file is a more stringent version of a *.gff*. file. The first eight fields in the *.gtf file are the same as the first eight fields in a *.gff file. The group field in the *.gtf file has been expanded into a list of attributes. Each attribute consists of a type/value pair. Attributes must end in a semicolon and be separated from any following attribute by exactly one space.

A *.gtf file is provided to the `bioscope.sh` command via the wt.splext.genegtf file parameter. The *.gtf is the file that defines the genes and transcripts in the reference genomes. Details about the *.gtf format can be obtained at

mblab.wustl.edu/GTF2.html

A *.gtf file containing the human genes in the UCSC Genome browser's RefGene table is available from

www.solidsoftwaretools.com

IMPORTANT! The *.gtf files that are available directly from the UCSC Genome Browser are not appropriate for this tool because they do not group features by gene_id. BioScope™ Software requires this file to group exons from the same gene with common values in the gene_id field in the attributes column.

BioScope™ Software provides a program, `bin/refgene2gtf.sh,` that you can use to convert the RefGene table (RefGene.txt) from the UCSC Genome Browser to a *.gtf format appropriate for use with BioScope™ Software WTA tools. The RefGene.txt file is available from

hgdownload.cse.ucsc.edu/goldenPath/hg18/database/

BioScope™ Software has only tested this program with the UCSC human RefGene.txt. In cases where a RefGene.txt file is not available, a custom script for preparing the *.gtf file is required.

Run Count Known Exons

This section explains how to run the Count Known Exons tool from the command line or the web interface.

Select the required input files

Before you can run the Count Known Exons tool you must know:

- The absolute path to the *.bam file.
- The absolute path to the Exon Reference(*.gtf) file.
- Changes to the default read length of 50, (optional).

Complete the prerequisites

1. Complete the applicable prerequisites described in Chapter 3, “Before you Begin” on page 35.
2. Login to the BioScope™ Software cluster. Change to the working directory and update the wt.ini file with information that applies to the Count Known Exons run that you want to initiate.
3. Complete the secondary whole transcriptome analysis on the primary data from the instrument.
4. Convert RefGene.txt to RefGene.gtf:

```
% refgene2gtf.sh =i refGene.txt -o refGene.gtf
```

Run Count Known Exons from the command line

Although several different software programs are involved in the experiment, a single command generates all of the related programs required to complete the experiment. The *.plan file that is specified in the command syntax controls the order in which BioScope™ Software runs the related programs.

1. Connect to the BioScope™ Software cluster and login with a user ID that has write privileges on all of the directories that BioScope™ Software uses when the tool runs.
2. At a command prompt, enter:
   ```
   bioscope.sh -l filename.log filename.plan
   ```

Do not log out of the BioScope™ Software cluster.

Check the run status from the command line

1. Navigate to the log directory that is defined in the wt.pe.counttag.ini file. For example, you might enter:
   ```
   cd /data/results/tertiary/output/log
   ```
2. Open `bioscope.`*`yyyymmddhhmmss`*`.log`.
3. Scroll to the end of the file.

The run is complete if you see an entry similar to:

```
15 Apr 2010 03:16:32,537  INFO [main] PluginJobManager:130 -
>>>> END of PluginJobManager >>>> date DURATION=4 minutes 33
secs

15 Apr 2010 03:16:32,537  INFO [main] EventTransportFactory:129
- Closing JMS connection and session
```

Run the Count Known Exons tool from the web interface

The instructions in this section assume the following system conditions:

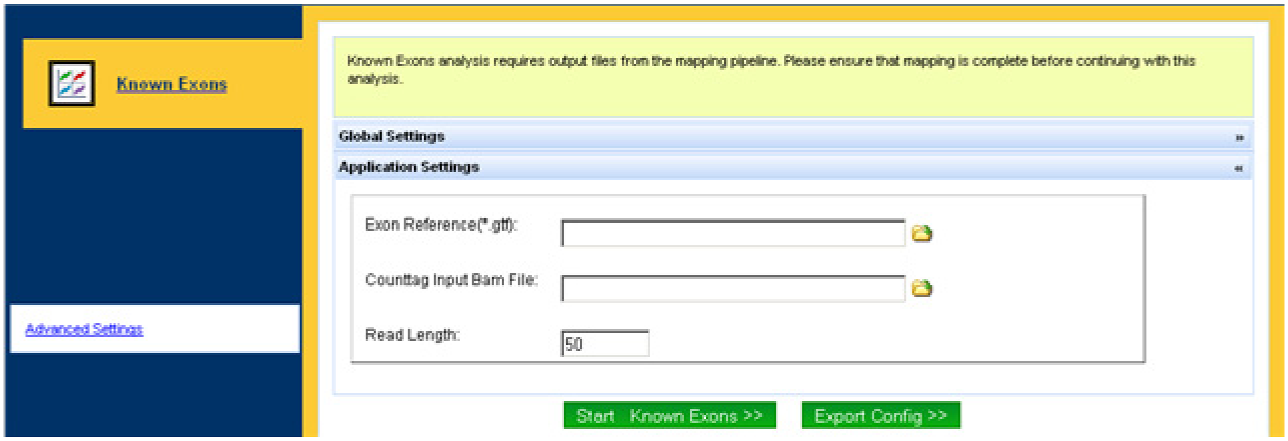
- The Java Messenger, Tomcat, and Apache services are running on the BioScope™ Software cluster.
- You are using Internet Explorer versions 6 or 7 or Mozilla 3.0.1.
- Mate-pair mapping is complete.

Launch a browser and enter the BioScope™ Software URL:
http://<hostname>:8080/bioscope

1. Click **Count Known Exons**.

The Count Known Exons page has two windows and one link (see Figure 20 on page 96).

- Global Settings
- Applications Settings
- Advanced Settings



Known Exons analysis requires output files from the mapping pipeline. Please ensure that mapping is complete before continuing with this analysis.

Exon Reference(*.gtf):

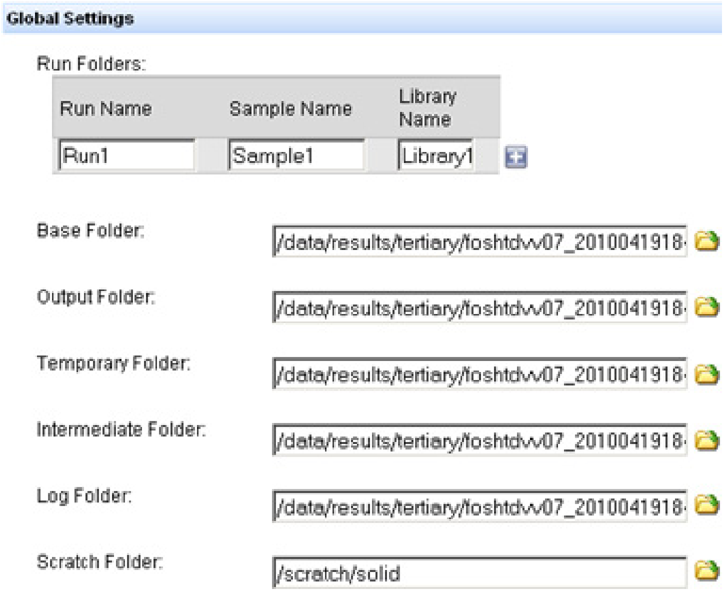
Counttag Input Bam File:

Read Length: 50

Figure 20 Count Known Exons Web page example

Global Settings description

The Global Settings window displays the default values for the folders that BioScope™ Software creates for the files that result from the Count Known Exons run (see Figure 21 on page 97).



Global Settings

Run Folders:

Run Name	Sample Name	Library Name
Run1	Sample1	Library1

Base Folder: /data/results/tertiary/foshtdw07_2010041918

Output Folder: /data/results/tertiary/foshtdw07_2010041918

Temporary Folder: /data/results/tertiary/foshtdw07_2010041918

Intermediate Folder: /data/results/tertiary/foshtdw07_2010041918

Log Folder: /data/results/tertiary/foshtdw07_2010041918

Scratch Folder: /scratch/solid

Figure 21 Count Known Exons Global Settings example

Customize the default folder structure (optional)

The folders store the results files generated by each Count Known Exons run. BioScope™ Software automatically creates the default folder structure for each Count Known Exons run:
`/data/results/tertiary/`*`headnode_yyyymmddhhmmss_x`*

Complete the following steps to change the default directory structure.

1. Click 📂 in the Base Folder field. The File Browser dialog appears.
2. In the **Look in** field, type the custom directory path, for example, `/home/data`
3. Click **Open**.
4. The folders reflect the updated directory structure.

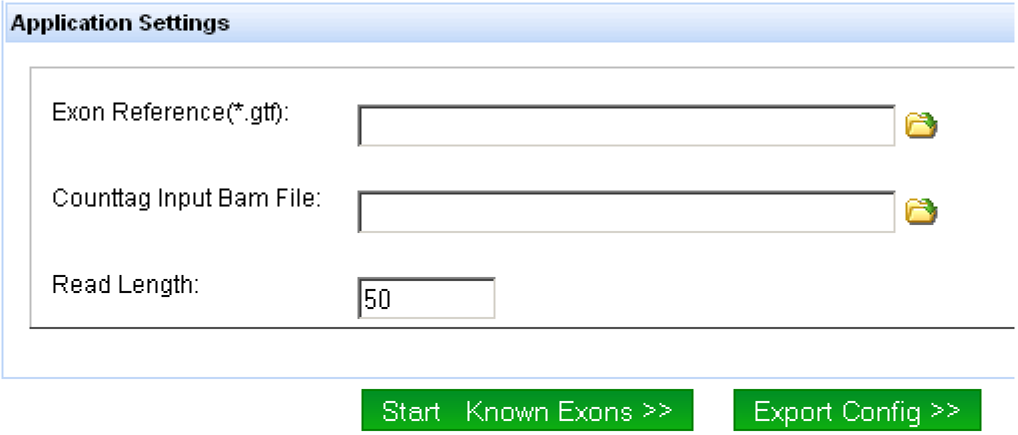
Note: If you change the default directory structure, the Output, Temporary, Intermediate, and Log folders become subdirectories of the Base Folder.

Advanced Settings description

Click **Advanced Settings** to view the current default values defined by BioScope™ Software for the Count Known Exons tool. Do *not* change any Advanced Settings unless instructed to by the BioScope™ Software administrator.

Application Settings description

In the Application Settings window (see Figure 22), you must define the absolute path to the *.gtf file. You must also define the absolute path to the Counttag Input Bam file. You have the option to modify the default Read Length value. You also start the Count Known Exons run from the Applications Settings window. The Export Config >> button is only used with the tool that processes barcoded libraries **(see** Appendix C, "Batch Analysis of Barcoded Library Data" on page 319).



Application Settings

Exon Reference(*.gtf):

Counttag Input Bam File:

Read Length: 50

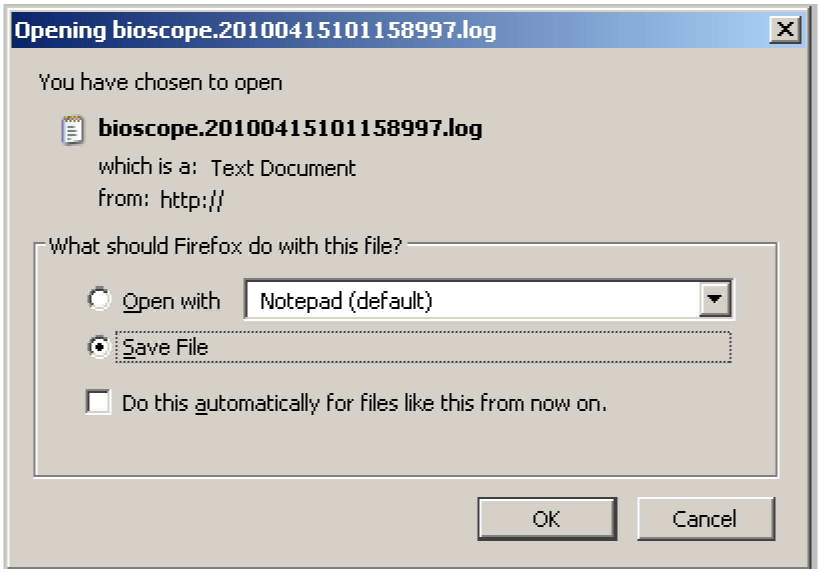
Figure 22 Count Known Exons Application Settings window

Start the Count Known Exons tool run

1. Click 📂 in the Exon Reference(*gtf) field. The File Browser window appears.
2. Define the absolute path to the *.bam file.
3. Click **Open**.
4. Optional: Update the Read Length value. Click Start Known Exons >> to start the run.
5. At the job submission dialog, click **OK** after you have verified the folder locations.

Check the status of the run from the web interface

1. Click **History** . The History window appears and the History Details table is displayed in the left pane. The History Details table shows the Time Created and Analysis Name for all runs performed on the BioScope™ Software cluster.
2. Scroll the History Details table and select the Counttags run, based on the data in the Time Created column.
3. Click **Download**.
   - Click **Open with** and click **OK** to view the log file in Notepad or select a different text editor.
   - Click **Save File** to copy the file to your workstation.



Opening bioscope.20100415101158997.log

You have chosen to open

bioscope.20100415101158997.log

which is a: Text Document
from: http://

What should Firefox do with this file?

○ Open with Notepad (default)

● Save File

□ Do this automatically for files like this from now on.

4. Scroll to the end of the file.

The run is complete if you see an entry similar to:

```
15 Apr 2010 03:16:32,537  INFO [main] PluginJobManager:130 -
>>>> END of PluginJobManager >>>> date DURATION=4 minutes 33
secs

15 Apr 2010 03:16:32,537  INFO [main] EventTransportFactory:129
- Closing JMS connection and session
```

7 Run the Create UCSC WIG File Tool

This chapter covers:

Create UCSC WIG File introduction

This tool converts the *.bam file that is created by the WT mapping and pairing tool into a *.wig file containing coverage data. Coverage is the number of reads covering a given genome stranded position.

Two .wig coverage files, one for each strand, are created in the output directory, for example: `/output/sam2wig/`. A *.wig file can be visualized in the UCSC Genome Browser (see Figure 28 on page 107

The preferred way to generate the *.wig file is with filter wt.counttag.min.mapq, which is set to 20 by default. The rationale for this threshold is that pairing QV 20 is a good uniqueness threshold for alignments. If the MAPQ filter is used, only the alignments above selected QV will contribute to coverage calculated in the *.wig file.

For information about the algorithm and other parameters associated with the tool, see "Whole Transcriptome Pipeline Concepts" on page 47.

GTF file format description

A *.gtf file is a more stringent version of a *.gff*. file. The first eight fields in the *.gtf file are the same as the first eight fields in a *.gff file. The group field in the *.gtf file has been expanded into a list of attributes. Each attribute consists of a type/value pair. Attributes must end in a semicolon and be separated from any following attribute by exactly one space.

A *.gtf file is provided to the `bioscope.sh` command via the wt.splext.genegtf file parameter. The *.gtf is the file that defines the genes and transcripts in the reference genomes. Details about the *.gtf format can be obtained at

mblab.wustl.edu/GTF2.html

A *.gtf file containing the human genes in the UCSC Genome browser's RefGene table is available from

www.solidsoftwaretools.com

IMPORTANT! The *.gtf files that are available directly from the UCSC Genome Browser are not appropriate for this tool because they do not group features by gene_id. BioScope™ Software requires this file to group exons from the same gene with common values in the gene_id field in the attributes column.

BioScope™ Software provides a program, `bin/refgene2gtf.sh,` that you can use to convert the RefGene table (RefGene.txt) from the UCSC Genome Browser to a *.gtf format appropriate for use with BioScope™ Software WTA tools. The RefGene.txt file is available from

hgdownload.cse.ucsc.edu/goldenPath/hg18/database/

BioScope™ Software has only tested this program with the UCSC human RefGene.txt. In cases where a RefGene.txt file is not available, a custom script for preparing the *.gtf file is required.

Prepare to run the Create UCSC WIG File tool

This section explains how to prepare to run the Create UCSC WIG File tool from the command line or the web interface.

Select the required input files

Before you can run the Create UCSC WIG File tool you must know:

- The absolute path to the *.bam file
- Changes to the default read length of 50, (optional)

Complete the prerequisites

1. Complete the applicable prerequisites described in Chapter 3, "Before you Begin" on page 35.
2. Login to the BioScope™ Software cluster. Change to the working directory and update the wt.ini file with information that applies to the Create UCSC WIG File run that you want to initiate.
3. Convert RefGene.txt to RefGene.gtf:

```
% refgene2gtf.sh =i refGene.txt -o refGene.gtf
```

4. Complete the secondary whole transcriptome analysis on the primary data from the instrument.

Run the Create UCSC WIG File from the command line

Although several different software programs are involved in the experiment, a single command generates all of the related programs required to complete the experiment. The *.plan file that is specified in the command syntax controls the order in which BioScope™ Software runs the related programs.

1. Connect to the BioScope™ Software cluster and login with a user ID that has write privileges on all of the directories that BioScope™ Software uses when the tool runs.
2. At a command prompt, enter:
   `bioscope.sh -l` *`filename`*`.log` *`filename`*`.plan`

Do not log out of the BioScope™ Software cluster.

Check the run status from the command line

1. Navigate to the log directory that is defined in the wt.pe.sam2wig.ini file. For example, you might enter:

   ```
   cd /data/results/tertiary/log
   ```

2. Open `bioscope.`*`yyyymmddhhmmss`*`.log`.
3. Scroll to the end of the file.

The run is complete if you see an entry similar to:

```
15 Apr 2010 03:16:32,537  INFO [main] PluginJobManager:130 -
>>>> END of PluginJobManager >>>> date DURATION=4 minutes 33
secs
```

```
15 Apr 2010 03:16:32,537  INFO [main] EventTransportFactory:129
- Closing JMS connection and session
```

Run the Create UCSC WIG File from the web interface

The instructions in this section assume the following system conditions:

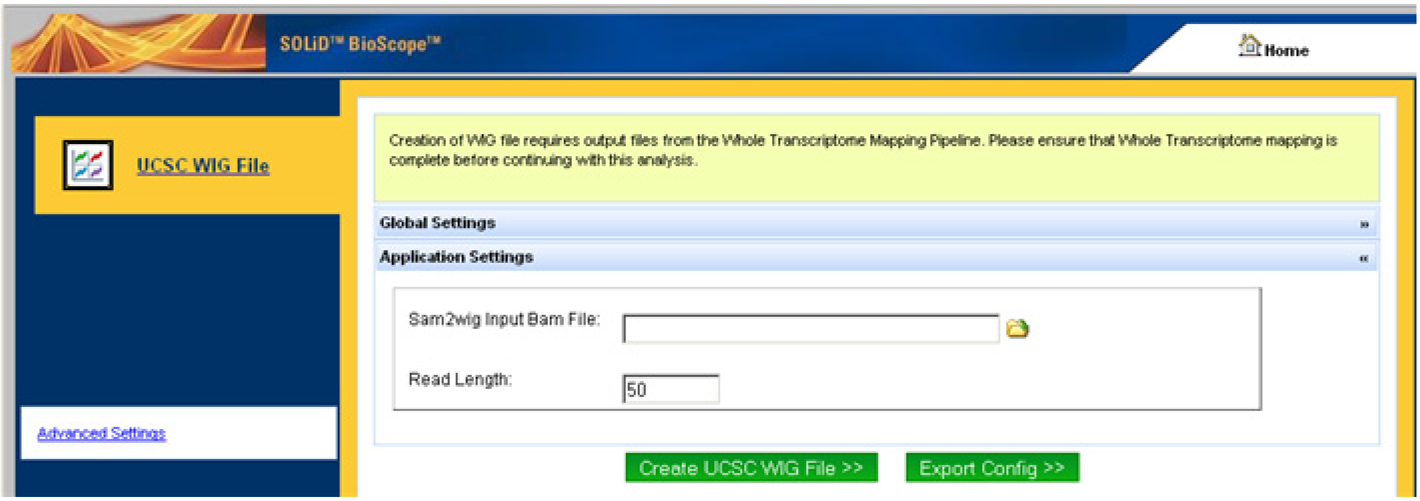
- The Java Messenger, Tomcat, and Apache services are running on the BioScope™ Software cluster.
- You are using Internet Explorer versions 6 or 7 or Mozilla 3.0.1.
- Mate-pair mapping is complete.

1. Launch a browser and enter the BioScope™ Software URL: http://<hostname>:8080/bioscope

2. Click **Create UCSC WIG File**.

The Create UCSC WIG File page has two windows and one link (see Figure 23 on page 104).

- Global Settings
- Applications Settings
- Advanced Settings



Creation of WIG file requires output files from the Whole Transcriptome Mapping Pipeline. Please ensure that Whole Transcriptome mapping is complete before continuing with this analysis.

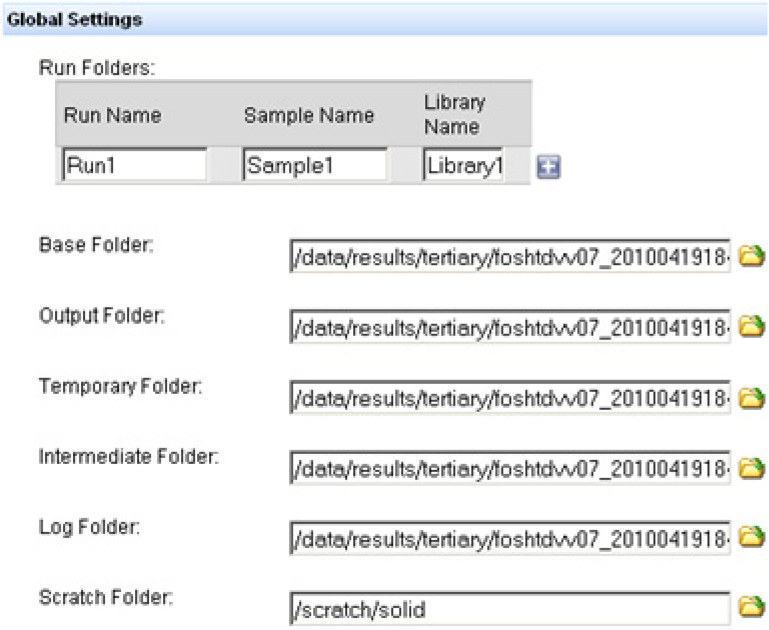
Sam2wig Input Bam File:

Read Length: 50

Figure 23 Create UCSC WIG File web page example

Global Settings description

The Global Settings window displays the default values for the folders that BioScope™ Software creates for the files that result from the Create UCSC WIG File run (see Figure 24 on page 105).



Global Settings

Run Folders:

Run Name	Sample Name	Library Name
Run1	Sample1	Library1

Base Folder: /data/results/tertiary/foshtdw07_2010041918

Output Folder: /data/results/tertiary/foshtdw07_2010041918

Temporary Folder: /data/results/tertiary/foshtdw07_2010041918

Intermediate Folder: /data/results/tertiary/foshtdw07_2010041918

Log Folder: /data/results/tertiary/foshtdw07_2010041918

Scratch Folder: /scratch/solid

Figure 24 Create UCSC WIG File Global Settings example

Customize the default folder structure (optional)

The folders store the results files generated by each Create UCSC WIG File run. BioScope™ Software automatically creates the default folder structure for each Create UCSC WIG File run:
`/data/results/tertiary/`*`headnode_yyyymmddhhmmss_x`*

Complete the following steps to change the default directory structure.

1. Click 📂 in the Base Folder field. The File Browser dialog appears.
2. In the **Look in** field, type the custom directory path, for example, `/home/data`
3. Click **Open**.
4. The folders reflect the updated directory structure.

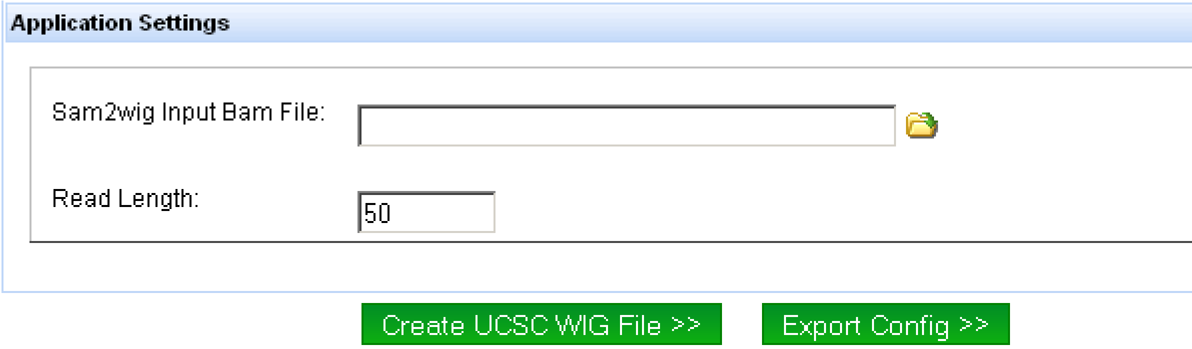
Note: If you change the default directory structure, the Output, Temporary, Intermediate, and Log folders become subdirectories of the Base Folder.

Advanced Settings description

Click **Advanced Settings** to view the current default values defined by BioScope™ Software for the Create UCSC WIG File tool. Do *not* change any Advanced Settings unless instructed to by the BioScope™ Software administrator.

Application Settings description

In the Application Settings window (see Figure 25), you must define the absolute path to the *.bam file. You also start the Create UCSC WUG File run from the Applications Settings window. The Export Config >> button is only used with the tool that processes barcoded libraries (see Appendix C, "Batch Analysis of Barcoded Library Data" on page 319).



Application Settings

Sam2wig Input Bam File:

Read Length: 50

Figure 25 Create UCSC WIG File Application Settings window

Start the Create UCSC WIG File tool run

1. Click 📂 in the Sam2wig Input Bam File field. The File Browser window appears.

2. Define the absolute path to the *.bam file.

3. Click **Open**.

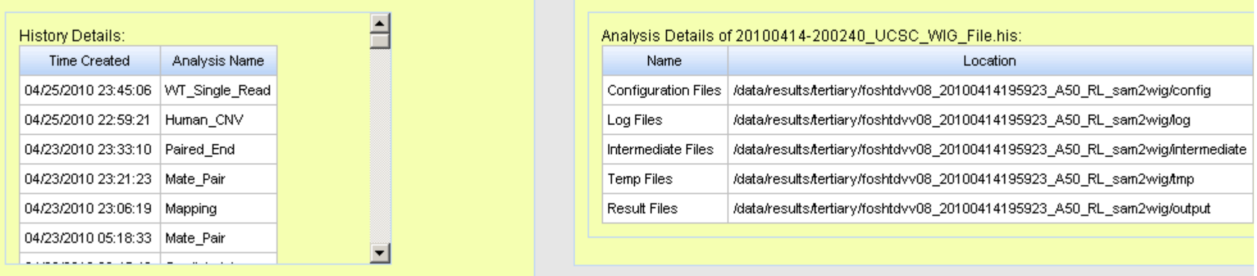
4. Optional: Update the Read Length value. Click Create UCSC WIG File >> to start the run.

5. At the job submission dialog, click **OK** after you have verified the folder locations.

Check the status of the run from the web interface

1. Click History . The History window appears and the History Details table is displayed in the left pane. The History Details table shows the Time Created and Analysis Name for all runs performed on the BioScope™ Software cluster.

2. Scroll the History Details table and select the UCSC_WIG_File run, based on the data in the Time Created column (see Figure 26).



History Details:

Time Created	Analysis Name
04/25/2010 23:45:06	WT_Single_Read
04/25/2010 22:59:21	Human_CNV
04/23/2010 23:33:10	Paired_End
04/23/2010 23:21:23	Mate_Pair
04/23/2010 23:06:19	Mapping
04/23/2010 05:18:33	Mate_Pair

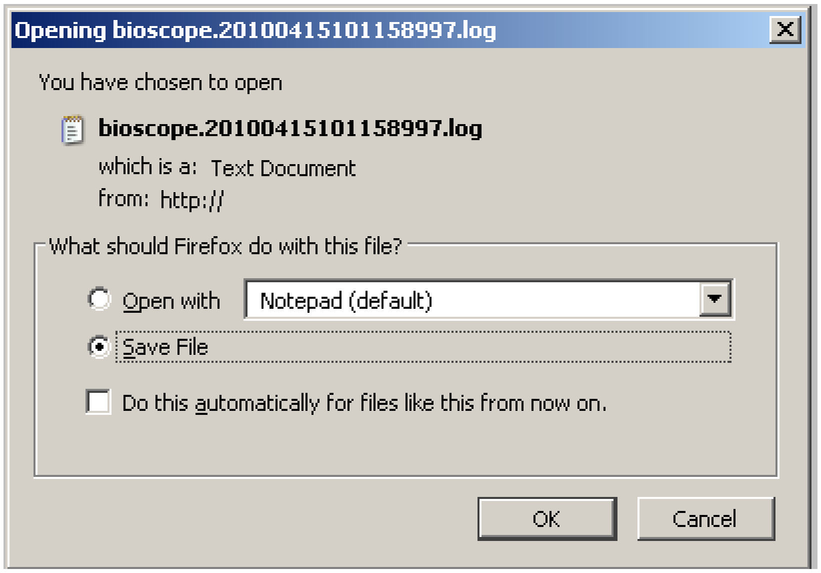
Analysis Details of 20100414-200240_UCSC_WIG_File.his:

Name	Location
Configuration Files	/data/results/tertiary/foshtdvv08_20100414195923_A50_RL_sam2wig/config
Log Files	/data/results/tertiary/foshtdvv08_20100414195923_A50_RL_sam2wig/log
Intermediate Files	/data/results/tertiary/foshtdvv08_20100414195923_A50_RL_sam2wig/intermediate
Temp Files	/data/results/tertiary/foshtdvv08_20100414195923_A50_RL_sam2wig/tmp
Result Files	/data/results/tertiary/foshtdvv08_20100414195923_A50_RL_sam2wig/output

Figure 26 History details and analysis details for Create UCSC WIG File tool run

3. Click **Download**.

- Click **Open with** and click **OK** to view the log file in Notepad or select a different text editor.
- Click **Save File** to copy the file to your workstation.



Opening bioscope.20100415101158997.log

You have chosen to open

bioscope.20100415101158997.log

which is a: Text Document
from: http://

What should Firefox do with this file?

○ Open with Notepad (default)

● Save File

□ Do this automatically for files like this from now on.

Figure 27 Log file download page example

4. Scroll to the end of the file.

The run is complete if you see an entry similar to:

```
15 Apr 2010 03:16:32,537  INFO [main] PluginJobManager:130 -
>>>> END of PluginJobManager >>>> date DURATION=4 minutes 33
secs

15 Apr 2010 03:16:32,537  INFO [main] EventTransportFactory:129
- Closing JMS connection and session
```

Create UCSC WIG File results file example

See Figure 28 for an example of WIG file output visualized in the UCSC genome browser. The tracks show the genomic coverage using the negative strand and positive strand specific wig files generated by the Bam2Wig tool (Max: 100 coverage).

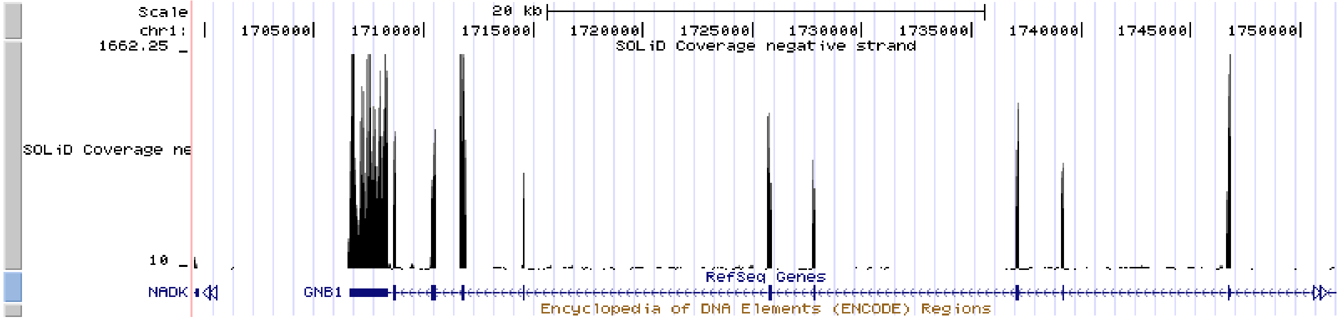


Figure 28 GNB1 gene (on negative strand) of human chromosome 1 displayed with UCSC genome browser custom tracks

8 Run the Find Splicing Fusion Tool

This chapter covers:

Introduction

A fusion junction is a section of transcribed RNA that maps to an exon from one gene followed by an exon from another gene. It can occur as the result of a translocation, deletion, or chromosomal inversion. A fusion junction excludes exon-to-exon boundaries that arise from alternative splicing for a gene.

GTF file format description

A *.gtf file is a more stringent version of a *.gff. file. The first eight fields in the *.gtf file are the same as the first eight fields in a *.gff file. The group field in the *.gtf file has been expanded into a list of attributes. Each attribute consists of a type/value pair. Attributes must end in a semicolon and be separated from any following attribute by exactly one space.

A *.gtf file is provided to the `bioscope.sh` command via the `wt.splext.genegtf` file parameter. The *.gtf is the file that defines the genes and transcripts in the reference genomes. Details about the *.gtf format can be obtained at

mblab.wustl.edu/GTF2.html

A *.gtf file containing the human genes in the UCSC Genome browser’s RefGene table is available from

www.solidsoftwaretools.com

IMPORTANT! The *.gtf files that are available directly from the UCSC Genome Browser are not appropriate for this tool because they do not group features by gene_id. BioScope™ Software requires this file to group exons from the same gene with common values in the gene_id field in the attributes column.

BioScope™ Software provides a program, `bin/refgene2gtf.sh`, that you can use to convert the RefGene table (RefGene.txt) from the UCSC Genome Browser to a *.gtf format appropriate for use with BioScope™ Software WTA tools. The RefGene.txt file is available from

hgdownload.cse.ucsc.edu/goldenPath/hg18/database/

BioScope™ Software has only tested this program with the UCSC human RefGene.txt. In cases where a RefGene.txt file is not available, a custom script for preparing the *.gtf file is required.

Prepare to run Find Splicing Fusion

This section explains how to run the Find Splicing Fusion tool from the command line or the web interface.

Select the required input files

Before you can run the Find Splicing Fusion tool you must know:

- The absolute path to the *.bam file.
- The absolute path to the Exon Reference(*.gtf) file.
- Changes to the default read length of 50, (optional).

Complete the prerequisites

1. Complete the applicable prerequisites described in Chapter 3, "Before you Begin" on page 35.
2. Login to the BioScope™ Software cluster. Change to the working directory and update the wt.ini file with information that applies to the Find Splicing Fusion run that you want to initiate.
3. Complete the secondary whole transcriptome analysis on the primary data from the instrument.
4. Convert RefGene.txt to RefGene.gtf:

```
% refgene2gtf.sh =i refGene.txt -o refGene.gtf
```

Run Find Splicing Fusion from the command line

Although several different software programs are involved in the experiment, a single command generates all of the related programs required to complete the experiment. The *.plan file that is specified in the command syntax controls the order in which BioScope™ Software runs the related programs.

1. Connect to the BioScope™ Software cluster and login with a user ID that has write privileges on all of the directories that BioScope™ Software uses when the tool runs.
2. At a command prompt, enter:
   `bioscope.sh -l` *`filename`*`.log` *`filename`*`.plan`

Do not log out of the BioScope™ Software cluster.

Check the run status from the command line

1. Navigate to the log directory that is defined in the wt.pe.juntionfinder.ini file. For example, you might enter:

   ```
   cd /data/results/tertiary/output/log
   ```

2. Open `bioscope.`*`yyyymmddhhmmss`*`.log`.
3. Scroll to the end of the file.

```
The run is complete if you see an entry similar to:
15 Apr 2010 03:16:32,537  INFO [main] PluginJobManager:130 -
>>>> END of PluginJobManager >>>> date DURATION=4 minutes 33
secs
15 Apr 2010 03:16:32,537  INFO [main] EventTransportFactory:129
- Closing JMS connection and session
```

Run the Find Splicing Fusion tool from the web interface

The instructions in this section assume the following system conditions:

- The Java Messenger, Tomcat, and Apache services are running on the BioScope™ Software cluster.
- You are using Internet Explorer versions 6 or 7 or Mozilla 3.0.1.

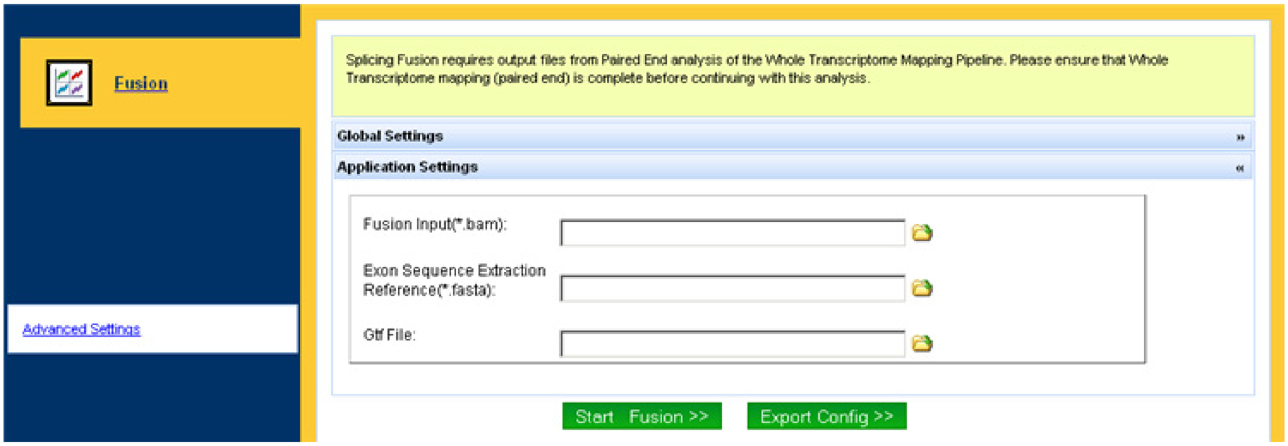
- Mate-pair mapping is complete.

1. Launch a browser and enter the BioScope™ Software URL:
   http://<hostname>:8080/bioscope

2. Click **Find Splicing Fusion**.

The Find Splicing Fusion page has two windows and one link (see Figure 29 on page 112).

- Global Settings
- Applications Settings
- Advanced Settings



Splicing Fusion requires output files from Paired End analysis of the Whole Transcriptome Mapping Pipeline. Please ensure that Whole Transcriptome mapping (paired end) is complete before continuing with this analysis.

Fusion Input(*.bam):

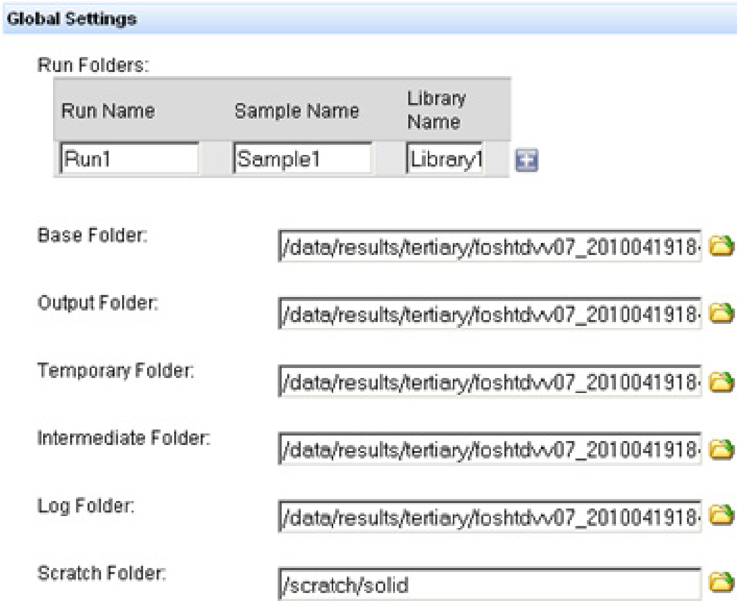
Exon Sequence Extraction Reference(*.fasta):

Gtf File:

Figure 29 Find Splicing Fusion web page example

Global Settings description

The Global Settings window displays the default values for the folders that BioScope™ Software creates for the files that result from the Find Splicing Fusion run (see Figure 30 on page 113).



Global Settings

Run Folders:

Run Name	Sample Name	Library Name
Run1	Sample1	Library1

Base Folder: /data/results/tertiary/foshtdvv07_2010041918·

Output Folder: /data/results/tertiary/foshtdvv07_2010041918·

Temporary Folder: /data/results/tertiary/foshtdvv07_2010041918·

Intermediate Folder: /data/results/tertiary/foshtdvv07_2010041918·

Log Folder: /data/results/tertiary/foshtdvv07_2010041918·

Scratch Folder: /scratch/solid

Figure 30 Find Splicing Fusion Global Settings example

Customize the default folder structure (optional)

The folders store the results files generated by each Find Splicing Fusion run. BioScope™ Software automatically creates the default folder structure for each Find Splicing Fusion run:
`/data/results/tertiary/`*`headnode_yyyymmddhhmmss_x`*

Complete the following steps to change the default directory structure.

1. Click 📂 in the Base Folder field. The File Browser dialog appears.
2. In the **Look in** field, type the custom directory path, for example, `/home/data`
3. Click **Open**.
4. The folders reflect the updated directory structure.

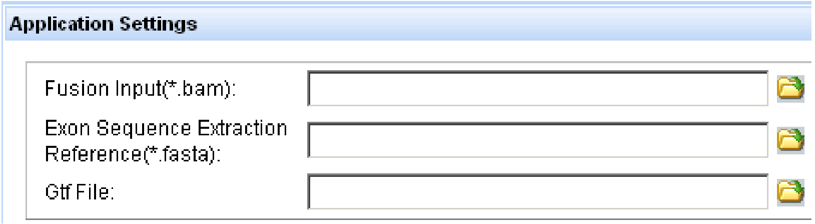
Note: If you change the default directory structure, the Output, Temporary, Intermediate, and Log folders become subdirectories of the Base Folder.

Advanced Settings description

Click **Advanced Settings** to view the current default values defined by BioScope™ Software for the Find Splicing Fusion tool. Do *not* change any Advanced Settings unless instructed to by the BioScope™ Software administrator.

Application Settings description

In the Application Settings window (see Figure 31), you must define the absolute path to the *.gtf file. You must also define the absolute path to the Counttag Input Bam file. You have the option to modify the default Read Length value. You also start the Find Splicing Fusion run from the Applications Settings window. The Export Config >> button is only used with the tool that processes barcoded libraries (see Appendix C, "Batch Analysis of Barcoded Library Data" on page 319).



Application Settings

Fusion Input(*.bam):

Exon Sequence Extraction Reference(*.fasta):

Gtf File:

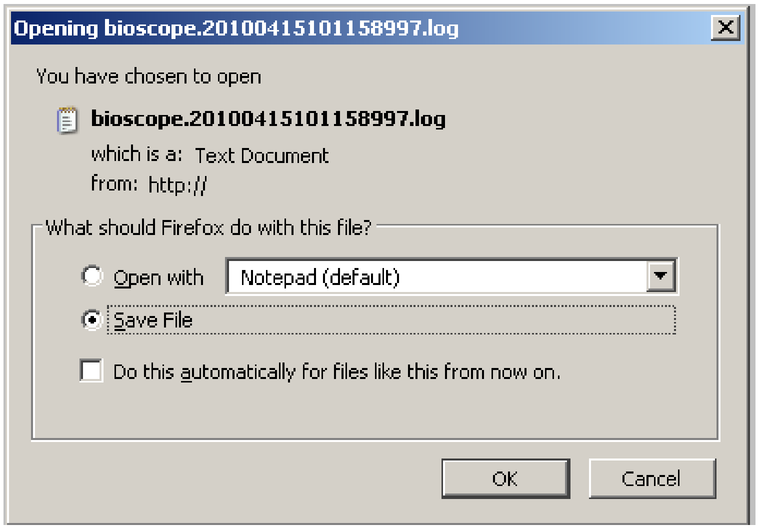
Figure 31 Find Splicing Fusion Application Settings window

Start the Find Splicing Fusion tool run

1. Click in the Exon Reference(*gtf) field. The File Browser window appears.
2. Define the absolute path to the *.bam file.
3. Click **Open**.
4. Optional: Update the Read Length value. Click **Start Fusion >>** to start the run.
5. At the job submission dialog, click **OK** after you have verified the folder locations.

Check the status of the run from the web interface

1. Click **History**. The History window appears and the History Details table is displayed in the left pane. The History Details table shows the Time Created and Analysis Name for all runs performed on the BioScope™ Software cluster.
2. Scroll the History Details table and select the Counttags run, based on the data in the Time Created column.
3. Click **Download**.
   - Click **Open with** and click **OK** to view the log file in Notepad or select a different text editor.
   - Click **Save File** to copy the file to your workstation.



Opening bioscope.20100415101158997.log

You have chosen to open

bioscope.20100415101158997.log

which is a: Text Document

from: http://

What should Firefox do with this file?

○ Open with Notepad (default)

● Save File

□ Do this automatically for files like this from now on.

4. Scroll to the end of the file.

The run is complete if you see an entry similar to:

```
15 Apr 2010 03:16:32,537  INFO [main] PluginJobManager:130 -
>>>> END of PluginJobManager >>>> date DURATION=4 minutes 33
secs
15 Apr 2010 03:16:32,537  INFO [main] EventTransportFactory:129
- Closing JMS connection and session
```

9 Run the Resequencing Mapping Tool

This chapter covers:

Mapping algorithm description

The next two sections explain the classic and current BioScope™ Software mapping algorithms.

Classic mapping

BioScope™ Software can perform mapping in classic mode.

Mapping capabilities and configuration features include:

- Multi-threaded mapreads for faster runtime.
- Seed-and-extend approach to mapping.
- A quality value is associated with each alignment. The quality value estimates the probability that the alignment is correct.
- Specification of local scratch space on available nodes.
- Temporary files that are handled to improve runtime performance.

Local mapping

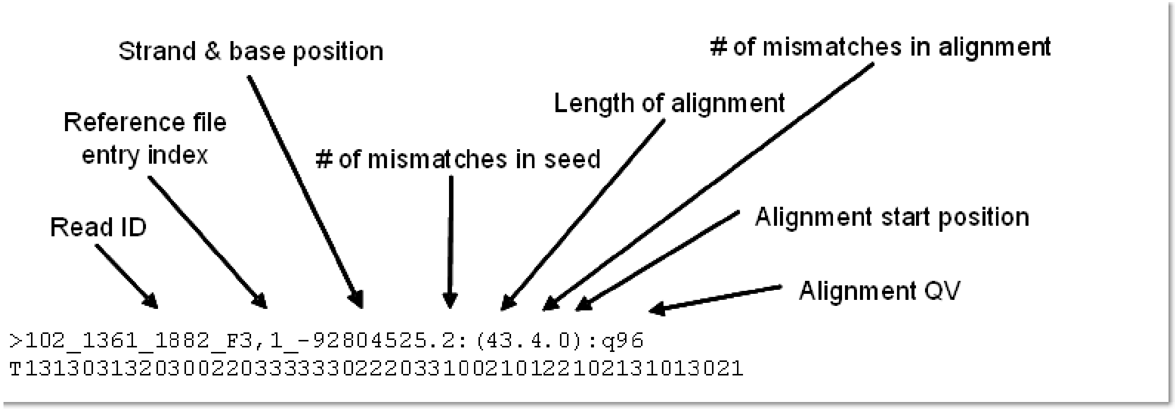
Alignment starts by locating short matches between a read and the reference sequence. With 50 base reads, the seed might be 25 bases long with up to two mismatches allowed. Extension only occurs when a read passes the initial seeding phase.

Extension can proceed in both directions, depending on the footprint of the seed within the read. During extension, each base match receives a score of +1, while mismatches get a default score of -2. You can configure the extension specification in the mapping.ini file.

Mapping pipeline example for a fragment run

You can set alignments all the way to the end of the genome while also keeping track of the ending scores of the alignment at each position. The mapping algorithm chooses the alignment with the highest score. The mapping algorithm chooses the shortest alignment if multiple possibilities have with the same score.

See Figure 32 for an explanation of the components in the mapping output file.



```
>102_1361_1882_F3,1_-92804525.2:(43.4.0):q96
T13130313203002203333330222033100210122102131013021
```

Figure 32 Mapping output explanation

The next paragraph refers to Figure 32.

The header shows that read 102_1361_1882_F3 has an alignment to chromosome 1 at position 92,804,525 on the reverse strand. The first color "1" after the primer base "T" should correspond to position 92,804,526 in the reference. The seed leading to this alignment had two mismatches. The fields between the parentheses reveal that the alignment is 43 colors long, has four mismatches, and starts at the beginning of the read. The alignment quality value is 96. The alignment length does not include the first color call. The mismatch number does include the first color call.

High-memory multi-schema

BioScope™ Software allows clusters with a minimum of 24 Gb of RAM to run multiple schemas on a single run. In the seed-mapping stage, 25 bases of each read are typically mapped to the genome. Two mismatches are allowed. The algorithm uses multiple discontinuous word schemas. Eight schemas and eight passes are needed to process the data for 25 bases with two mismatches.

In a cluster that has a large amount of available memory, two or more schemas can run at the same time. The mapping program decides how many schemas to run in a pass, depending on the size of the reference sequence and the available memory on the machine. The decision-making feature of the mapping program can reduce the time required for mapping because it reduces the disk I/O required for temporary results between passes.

Running the mapping analysis with different memory settings might yield slightly different results. When running multiple schemas in a pass, the order of the hits being found might change. The result remains the same for reads with less than *z* hits.

Multiple anchors in the same run

Mapping is often run multiple times with different anchor positions to increase sensitivity. For example, for 50 long-reads you might use 25 bases as the anchor. For the reads that have no hits, bases 16 to 40 are used as an anchor for the second run. For both runs, two mismatches are allowed in the anchor region. If the length of the anchors in the two runs are identical, and two mismatches are allowed, you can run the two steps together. Running the two steps together generally finds more hits. In previous versions of BioScope™ Software, the reads that had hits using the first anchor were not mapped using the second anchor. Now, both anchors are used simultaneously.

Using multiple anchors simultaneously results in is faster runtime because the intermediate steps of merging results and moving data are unnecessary. However, the results will differ depending on whether you run multiple anchors separately or together.

Note: Multi-anchor and non-multi-anchor runs use the same amount of RAM.

mapping.ini file example

The following section shows a typical example of a mapping.ini file and a small.indel.frag.ini file. For a description of the mapping.ini file parameters, see Table 19 on page 121. The small indel fragment parameters are described in Table 20 on page 124.

The parameter that is highlighted in *italics* in the following table must be changed for every mapping run. The parameters that are highlighted in **bold** need to be verified to make sure they are appropriate for the run. In most cases, all other parameter settings can remain as they are for each run.

IMPORTANT! Before you begin a mapping run, you must verify the settings for each parameter that is highlighted in **bold** in the mapping.ini file example below.

```
# If not specified or set to 1, clean up all intermediate files
# and temp folders afterwards
# Set to 0 for debugging
##pipeline.cleanup.middle.files = 0
##job.cleanup.temp.files = 0

# Global settings for the pipeline run
# Not required if only running mapping pipeline
run.name = test
sample.name = S1
primer.set = F3

# The location of reference file with full path
reference = /data/results/RegressionDriver/CaseManager/
knownData/validatedReference/genomes/
DH10B_WithDup_FinalEdit_validated.fasta

# The default length of read to use when running classic mapping
read.length = 50

# The default mismatch level if running classic mapping
# When not running classic mapping, this value is only used in
final match file name
mismatch.level = 2

# The directory where bioscope.sh command is executed
base.dir = .
# The directory for all outputs
output.dir = ${base.dir}/test

#
#   mapping pipeline
#
# Always set to 1 if running mapping pipeline
mapping.run = 1
# The location where the read file (*.csfasta) is located
mapping.tagfiles.dir = ${base.dir}/reads
# The output folder for mapping result file (*.ma)
mapping.output.dir = ${output.dir}/s_mapping
# Whether or not to run classic mapping
# Set to 1 to turn on classic mapping
mapping.run.classic = 0

# The length of read to use when running classic mapping
# By default use ${read.length}
##mapping.classic.anchor.length = ${read.length}
# The number of mismatches allowed when running classic mapping
# By default use ${mismatch.level}
##mapping.classic.mismatch = ${mismatch.level}
# Specifies a negative score for mismatch which is used in local
# alignment mode.  When this is set to a non-negative number,
the
# local mode is turned off and the output format of hits remains
# the same as that in V3.  Only set to a non-negative number when
# running classic mapping.
mapping.mismatch.penalty = -2.0
```

Mapping parameters

Table 19 Mapping parameter description

Parameter name	Default value	Description
Mandatory parameters		
mapping.run	1	Whether or not to run the mapping tool. Enter 0 if you do not want to run the tool.
mapping.output.dir	—	The output directory for the mapping pipeline output files.
Optional parameters		
mapping.repetitive.dir	—	The subfolder where the repetitive *.ma and *.csfasta files were written.
mapping.run.multithread	true	Whether or not to run mapreads in multithread mode. The default value is true when the parameter is not specified.
mapping.np.per.node	8	The number of processors per node use for mapping. The read file will be divided into the number of chunks specified for this parameter, and the chunks will be passed to mapreads.
mapping.number.of.nodes	3	The number of available nodes. The read file will be divided into the number of chunks specified for this parameter, and then further divided into the number of chunks specified in ${mapping.np.per.node}.
scratch.dir	/scratch/solid	The scratch folder location.
output.dir	./outputs	The output results directory.
mapping.output.dir	$ {output.dir}/s_mapping	The output folder where the *.ma files are placed.
mapping.tagfiles.dir	./reads	The folder where the *.csfasta file is placed.
mapping.np.per.node	8	The number of processors per node. The read file will be divided into this number of chunks and passed to mapreads.
mapping.number.of.nodes	3	The number of nodes available. The read file will be divided into the number of chunks specified for this parameter, and then further divided into number of chunks in the parameter specified for ${mapping.np.per.node}.
mapping.min.reads	—	The minimum number of reads the *.csfasta file should have for a read split to happen.

Table 19 Mapping parameter description *(continued)*

Parameter name	Default value	Description
mapping.memory.size	—	The total memory, in gigabytes, that is available for map reads. Note: The default value can be set during installation by selecting the lowest memory size available across the nodes in the cluster.
reference	—	The full path to the reference file.
read.length	25	The default value for read length if running classic mapping.
mismatch.level	6	The default value of the number of mismatches allowed when running classic mapping.
matching.max.hits	100	Defines the maximum number of best hits found in mapping.
mapping.write.sequence	1	Whether to write read sequences to the final *.ma file. Enter 0 if you do not want to write read sequences to the final *.ma file.
mapping.valid.adjacent	0	Penalize adjacent mismatches. Enter 1 to only count consistent adjacent mismatches as 1. Leave the default setting of 0 to count adjacent mismatches as 2.
mask.positions	—	An array of integers that indicate the positions in the read sequence that will be excluded in mapping. Leaving the parameter blank results in no masking.
mapping.schema.file	—	The schema file with the full path used in mapping.
matching.use.iub.reference	0	Whether or not to support reference sequences with IUB codes.matching.use.iub.reference. Enter 1 if you do not want to support reference sequences with IUB codes.matching.use.iub.reference.
mapping.run.classic	0	Whether or not to run classic mapping. Enter 1 if you do not want to run classic mapping.
mapping.classic.anchor.length	—	The length of read to use when running classic mapping. If you do not enter a value, the default is taken from the value defined in the read.length parameter.
mapping.classic.mismatch	—	The number of mismatches allowed when running classic mapping. If you do not enter a value, the default is taken from the value defined in the mismatch.level parameter.
mapping.hits.lower.limit	1	The lower limit of the number of hits. The value of the parameter is used to determine the branch that a matched read goes to during iterative mapping.
mapping.hits.upper.limit	100	The upper limit of number of hits. The value of the parameter is used to determine the branch that a matched read goes to during iterative mapping.
mapping.scheme.unmapped	—	Specifies a comma-separated list of anchorLength.mismatchAllowed.anchorStart data for unmapped reads.

Table 19 Mapping parameter description *(continued)*

Parameter name	**Default value**	**Description**
mapping.scheme.unmapped.25	25.2.0	Specifies a comma-separated list of anchorLength.mismatchAllowed.anchorStart data for unmapped reads of the length from 25 to 35.
mapping.scheme.repetitive.25	—	A comma-separated list of anchorLength.mismatchAllowed.anchorStart data for repetitive reads of length from 25 to 35.
mapping.scheme.unmapped.35	30.3.0	A comma-separated list of anchorLength.mismatchAllowed.anchorStart data for unmapped reads of the length from 35 to 50.
mapping.scheme.repetitive.35	empty	A comma-separated list of anchorLength.mismatchAllowed.anchorStart data that for repetitive reads of the length from 35 to 50.
mapping.scheme.unmapped.50	25.2.0, 25.2.15	A comma-separated list of anchorLength.mismatchAllowed.anchorStart data for unmapped reads of length greater or equal to 50.
mapping.scheme.repetitive.50	—	A comma-separated list of anchorLength.mismatchAllowed.anchorStart data for repetitive reads of length greater or equal to 50.
mapping.qual.error.rate	0.2	An estimate of the sequencing error rate.
mapping.qual.bvalue	1.0	An estimate of percentage of genome unique at length L with one mismatch. In humans, ten percent of positions match to somewhere else at 50.1.
mapping.qual.pvalue	1	Whether or not the *.ma file uses the multi-contig format. The value is always 1, which is the multi-contig format contig_pos.mm in BioScope™ Software.
mapping.qual.filter.cutoff	0	The minimum local score [0 to 100] of unique hits to filter out.
Temporary files and folders to keep		
pipeline.cleanup.middle.files	1	Whether or not to delete intermediate files generated in the mapping pipeline. Enter 0 to keep the intermediate files generated in the mapping pipeline.
job.clean.temp.files	1	Whether or not to delete intermediate files from split and gather. Enter 0 to keep the intermediate files from split and gather.
Mapping stats parameters		
clear.zone	5	The threshold to decide whether a read is mapped uniquely in the reference.
mapping.mismatch.penalty	-2.0	A negative score for mismatch which is used in local alignment mode. When this is set to a positive number, the local mode is turned off, and the output format of hits remains the same as the output format in SOLiD™ v3.0.
mapping.output.dir	=${output.dir}/s_mapping	The full path to the location where the mapping pipeline result is located.

Table 19 Mapping parameter description *(continued)*

Parameter name	Default value	Description
mapping.stats.output.file	=${output.dir}/s_mapping/ mapping-stats.txt	The full path and file name of the mapping.stats file generated by the mapping tool.

Determining gap alignments

Gap alignments are performed in Bioscope™ Software using the small indel frag tool. An algorithmic description of the small indel frag tool is found in Chapter 15, "Run the Find Small InDels Tool" on page 263.

The following section shows parameters used in a small.indel.frag.ini file. These parameters are explained in Table 20 on page 124.

```
small.indel.frag.run=1
base.dir = .
output.dir = ${base.dir}/../../../outputs

#       small.indel.frag.match                 .ma (match) file
where indels are to be found
small.indel.frag.match=${output.dir}/F3/s_mapping/
test_S1_F3.csfasta.ma

#       small.indel.frag.cmap                  CMAP file
(Chromosome mapping)
cmap=/some/path/to/cmap/human.cmap

#      small.indel.frag.output.dir          Results directory.
Default is smallindelfrag
small.indel.frag.output.dir=${output.dir}/fragGapAligner
```

Table 20 gives the parameters for small indel fragment mapping.

Table 20 Mapping parameters for small indel fragment runs

Parameter name	Default value	Description
Mandatory input parameters		
small.indel.frag.match		.ma (match) file
cmap		CMAP file (chromosome mapping)
Output directory		
small.indel.frag.output.dir	smallindelfrag/	Output directory
Algorithm parameters		

Parameter name	Default value	Description
small.indel.frag.indel.preset	1	Presets for indel parameters. Valid values: 1,3,4,5 1. Deletions to 11, insertions to 3 3. Insertions from 4 to 14 (not validated) 4. Insertions from 15 to 20 (not validated) 5. Longer deletions from 12 to 500 (not validated)
small.indel.frag.indel.parameters	D=11,I=4,d=13,i=10	Indel parameters
small.indel.frag.error.indel	3	Error total for indel finding
small.indel.frag.min.non.matched.length	10	Minimum non-mapped length for mapped reads
Other parameters		
small.indel.frag.qual		.qual (base quality) file (not needed if running maToBam afterwards)
processors.per.node.request	Set during install	Number of processing cores per node
memory.request	Set during install	Memory in megabytes per node
scratch.dir	/scratch/solid	Scratch directory
small.indel.frag.job.script.dir	smallindelfrag-job-dir	The job scripts directory
small.indel.frag.log.dir	smallindelfrag-log-dir/	Tool log files
small.indel.frag.intermediate.dir	intermediate-dir/	Intermediate files
pipeline.cleanup.middle.files	1	Set to 0 to keep intermediate files

Prepare to run the Map Data tool

Select the required input files

The type of input files required depends on the type of library you are mapping:

Inputs required to map fragment data

You must know the following information to map fragment data:

- The absolute path to the following files:
  - Multi-FASTA Reference File (*.fasta)
  - Reads File (*.csfasta)
  - Quality Value File(*.qual)
  - CMap file (for Small Indel fragment)
- The Primer Set (legal values: F3 or R3 or F5-P2 or F5-BC)
- The Read Length

Inputs required to map mate-pair data

You must know the following information to map mate-pair data:

- The absolute path to the following files:
  - Multi-FASTA Reference File (*.fasta)
  - F3 Reads File(*.csfasta)
  - F3 Quality Value File (*.qual)
  - R3 Reads File(*.csfasta)
  - R3 Quality Value File(*qual)
- The F3 Read Length
- The R3 Read Length

Inputs required to map paired-end data

You must know the following information to map paired-end data:

- The absolute path to the following files:
  - Multi-FASTA Reference File (*.fasta)
  - F3 Reads File(*.csfasta)
  - F3 Quality Value File (*.qual)
  - F5 Reads File(*.csfasta)
  - F5 Quality Value File(*qual)
- The Primer Set (legal values: F3,F5-P2 or F3,F5-BC)
- The F3 Read Length
- The F5 Read Length

Complete the prerequisites

1. Complete the applicable prerequisites described in Chapter 3, “Before you Begin” on page 35.

2. Login to the BioScope™ Software cluster. Change to the working directory and update the mapping.ini file with information that applies to the run. See “mapping.ini file example” on page 119.

Run the Map Data tool from the command line

Although several different software programs are involved in the experiment, a single command generates all of the related programs required to complete the experiment. The *.plan file that is specified in the command syntax controls the order in which BioScope™ Software runs the related programs.

Start the run

1. Connect to the BioScope™ Software cluster and login with a user ID that has write privileges on all of the directories that BioScope™ Software uses when the tool runs.

2. At a command prompt, enter:
   `bioscope.sh -l` *`filename`*`.log` *`filename`*`.plan`

Do not log out of the BioScope™ Software cluster.

Check the run status from the command line

1. Navigate to the log directory that is defined in the mapping.ini file. For example, you might enter:

   ```
   cd /data/results/tertiary/mapping/log
   ```

2. Open `bioscope.`*`yyyymmddhhmmss`*`.log`.

3. Scroll to the end of the file.

The run is complete if you see an entry similar to:

```
15 Apr 2010 03:16:32,537  INFO [main] PluginJobManager:130 -
>>>> END of PluginJobManager >>>> date DURATION=4 minutes 33
secs

15 Apr 2010 03:16:32,537  INFO [main] EventTransportFactory:129
- Closing JMS connection and session
```

Run the Map Data tool from the web interface

The instructions in this section assume the following system conditions:

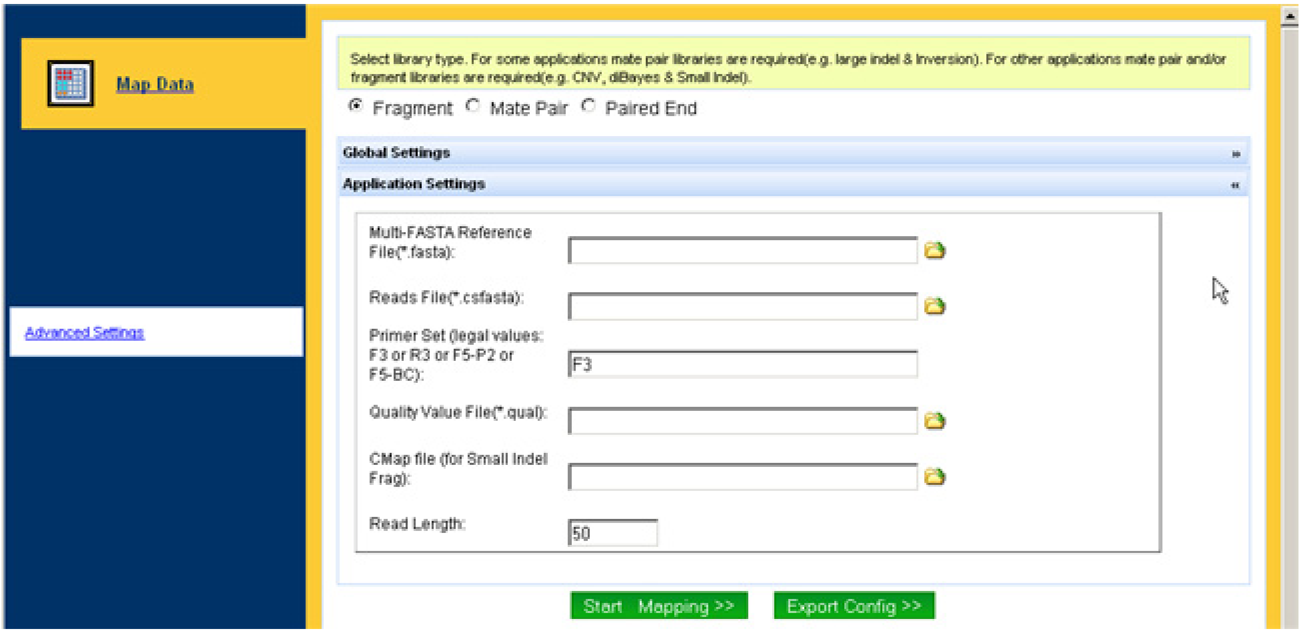
- The Java Messenger, Tomcat, and Apache services are running on the BioScope™ Software cluster.
- You are using Internet Explorer versions 6 or 7 or Mozilla 3.0.1.

1. Launch a browser and enter the BioScope™ Software URL:
   http://<hostname>:8080/bioscope

2. Click **Map Data**.

The Find Map Data page has two windows and one link (see Figure 33).

- Global Settings
- Applications Settings
- Advanced Settings



Select library type. For some applications mate pair libraries are required(e.g. large indel & Inversion). For other applications mate pair and/or fragment libraries are required(e.g. CNV, diBayes & Small Indel).

● Fragment ○ Mate Pair ○ Paired End

Global Settings

Application Settings

Multi-FASTA Reference File(*.fasta):

Reads File(*.csfasta):

Primer Set (legal values: F3 or R3 or F5-P2 or F5-BC): F3

Quality Value File(*.qual):

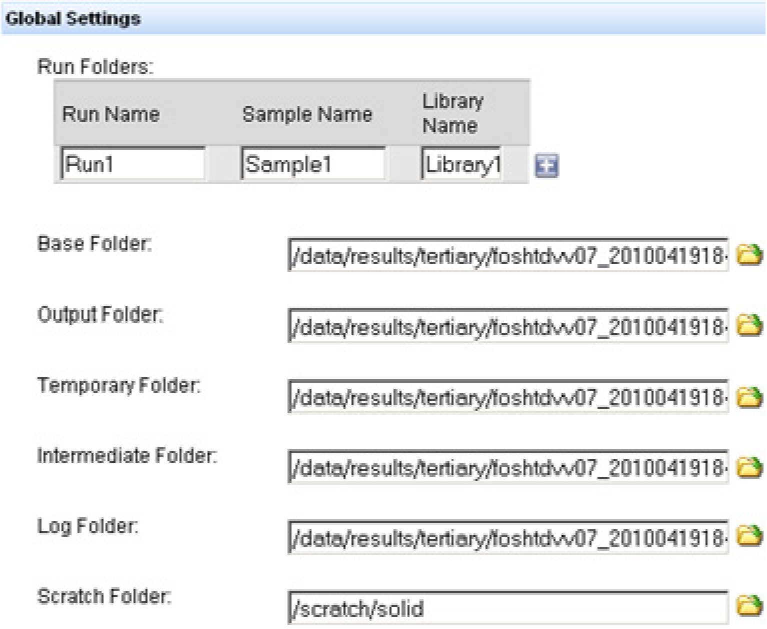
CMap file (for Small Indel Frag):

Read Length: 50

Figure 33 Map Data Web page example

Global Settings description

The Global Settings section displays the default values for the folders that BioScope™ Software creates for the files that result from the Map Data run (see Figure 34). The section also has fields where you can enter the Run Name, Sample Name, and Library Name of the primary data that was exported to BioScope™ Software from the instrument.



Global Settings

Run Folders:

Run Name	Sample Name	Library Name
Run1	Sample1	Library1

Base Folder: /data/results/tertiary/foshtdvw07_2010041918·

Output Folder: /data/results/tertiary/foshtdvw07_2010041918·

Temporary Folder: /data/results/tertiary/foshtdvw07_2010041918·

Intermediate Folder: /data/results/tertiary/foshtdvw07_2010041918·

Log Folder: /data/results/tertiary/foshtdvw07_2010041918·

Scratch Folder: /scratch/solid

Figure 34 Map Data Global Settings section example

Customize the default folder structure (optional)

The folders store the results files generated by each Map Data run. BioScope™ Software automatically creates the default folder structure for each Map Data run: `/data/results/tertiary/`*`headnode_yyyymmddhhmmss_x`*

Complete the following steps to change the default directory structure.

1. Click 📂 in the Base Folder field. The File Browser dialog appears.
2. In the **Look in** field, type the custom directory path, for example, `/home/data`
3. Click **Open**.
4. The folders reflect the updated directory structure.

Note: If you change the default directory structure, the Output, Temporary, Intermediate, and Log folders become subdirectories of the Base Folder.

Update the Run Folder settings (optional)

You can accept the default values in the Run Name, Sample Name and Library Name fields. In this context, "run" refers to the primary data that was exported to BioScope™ Software from the instrument.

To change the default values for the Run Folders:

1. Enter the updated run name in the Run Name field.
2. Enter the updated sample name in the Sample Name field.
3. Enter the updated library name in the Library Name field.
4. Optional: Click ⊞ to add a row for a second run folder.
5. Optional: Enter a Run Name, a Sample Name and a Library Name in the new row.

Advanced Settings description

Click **Advanced Settings** to view the current default values defined by BioScope™ Software for the Map Data tool. Do *not* change any Advanced Settings unless instructed to by the BioScope™ Software administrator.

Application Settings description

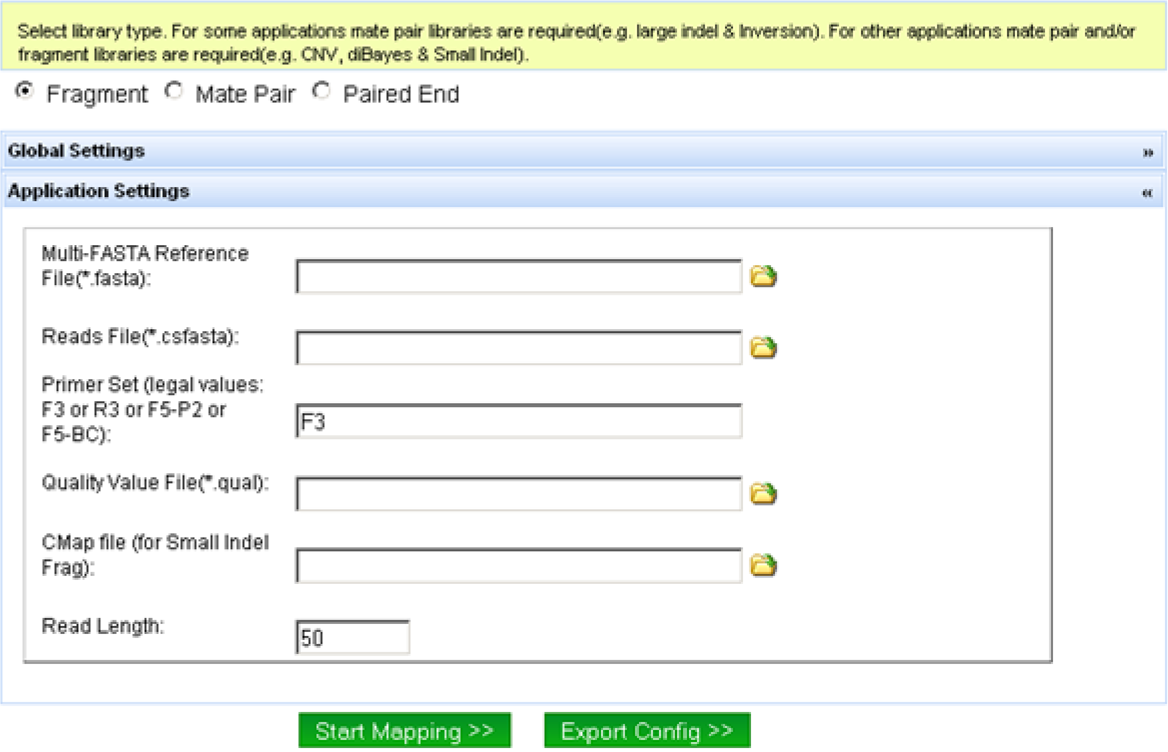
In the Application Settings window (see Figure 37 on page 133), you must select the data type of the library that you want to map. Depending on the data type, you enter different parameters. You also start the Map Data run from the Applications Setting window. The Export Config >> button is only used with the tool that processes barcoded libraries (see Appendix C, "Batch Analysis of Barcoded Library Data" on page 319).

Start the Map Fragment data tool run

If you are mapping fragment data, you must enter the Read Length and define the absolute path to the following files (see Figure 35 on page 130):

- Multi-FASTA Reference File (*.fasta)
- Reads File (*.csfasta)
- Primer Set (legal values: F3 or R3 or F5-P2 or F5-BC)

- Quality Value File(*.qual)
- CMap file (for Small Indel fragment)



Select library type. For some applications mate pair libraries are required(e.g. large indel & Inversion). For other applications mate pair and/or fragment libraries are required(e.g. CNV, diBayes & Small Indel).

● Fragment ○ Mate Pair ○ Paired End

Multi-FASTA Reference File(*.fasta):

Reads File(*.csfasta):

Primer Set (legal values: F3 or R3 or F5-P2 or F5-BC): F3

Quality Value File(*.qual):

CMap file (for Small Indel Frag):

Read Length: 50

Figure 35 Application Settings to map fragment data

1. Click **Fragment**.
2. Click 📂 in the Multi-FASTA Reference File(*.fasta) field. The File Browser window appears.
3. Define the directory path to the *.fasta file.
4. Click **Open**.
5. Click 📂 in the Reads File(*.csfasta) field. The File Browser window appears.
6. Define the directory path to the *.csfasta file.
7. Click **Open**.
8. Enter a value for the Primer Set.
9. Click 📂 in the Quality Value File (*.qual) field.
10. Define the directory path to the *.qual file.
11. Click **Open**.

12. Click 📂 in the CMap file (for Small Indel Frag) field.

13. Define the directory path to the CMap file.

14. Click **Open**.

15. Enter the length of the read in Read Length.

16. Click Start Mapping >> to start the run.

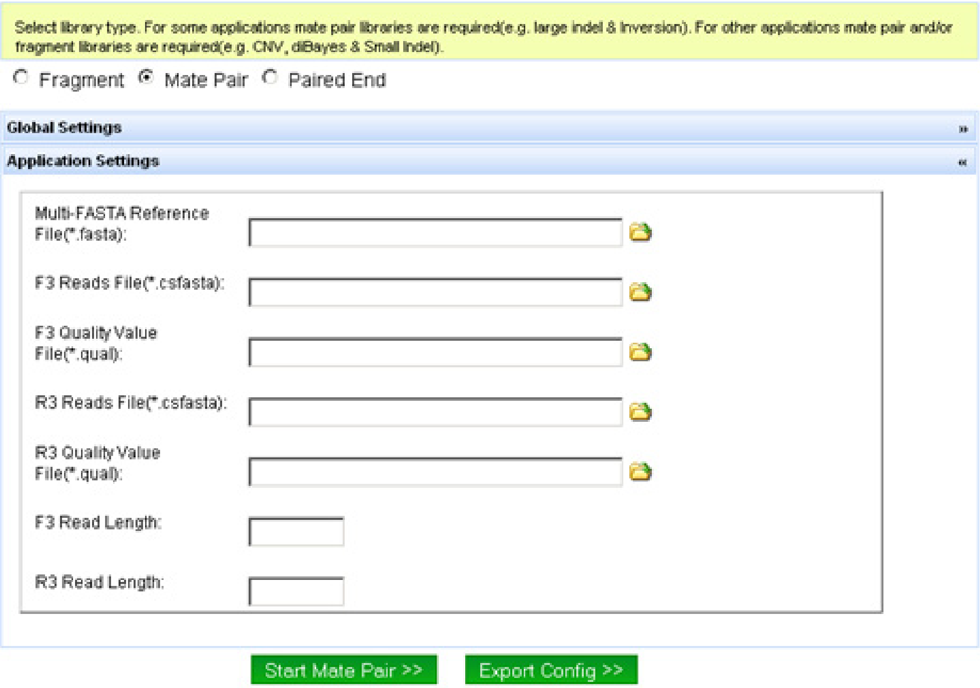
17. At the job submission dialog, click **OK** after you have verified the folder locations.

See "Check the status of the run from the web interface" on page 134 to view the status of the mapping run.

Start the Map Mate-Pair data tool run

If you are mapping mate-pair data, you must enter the F3 and R3 read lengths, and define the absolute path to the following files (see Figure 36):

- Multi-FASTA Reference File (*.fasta)
- F3 Reads File(*.csfasta)
- F3 Quality Value File (*.qual)
- R3 Reads File(*.csfasta)
- R3 Quality Value File(*qual)



Select library type. For some applications mate pair libraries are required(e.g. large indel & Inversion). For other applications mate pair and/or fragment libraries are required(e.g. CNV, diBayes & Small Indel).

○ Fragment ● Mate Pair ○ Paired End

Global Settings »

Application Settings «

Multi-FASTA Reference File(*.fasta):

F3 Reads File(*.csfasta):

F3 Quality Value File(*.qual):

R3 Reads File(*.csfasta):

R3 Quality Value File(*.qual):

F3 Read Length:

R3 Read Length:

Figure 36 Application Settings to map mate-pair data

1. Click **Mate Pair.**

2. Click 📂 in the Multi-FASTA Reference File(*.fasta) field. The File Browser window appears.

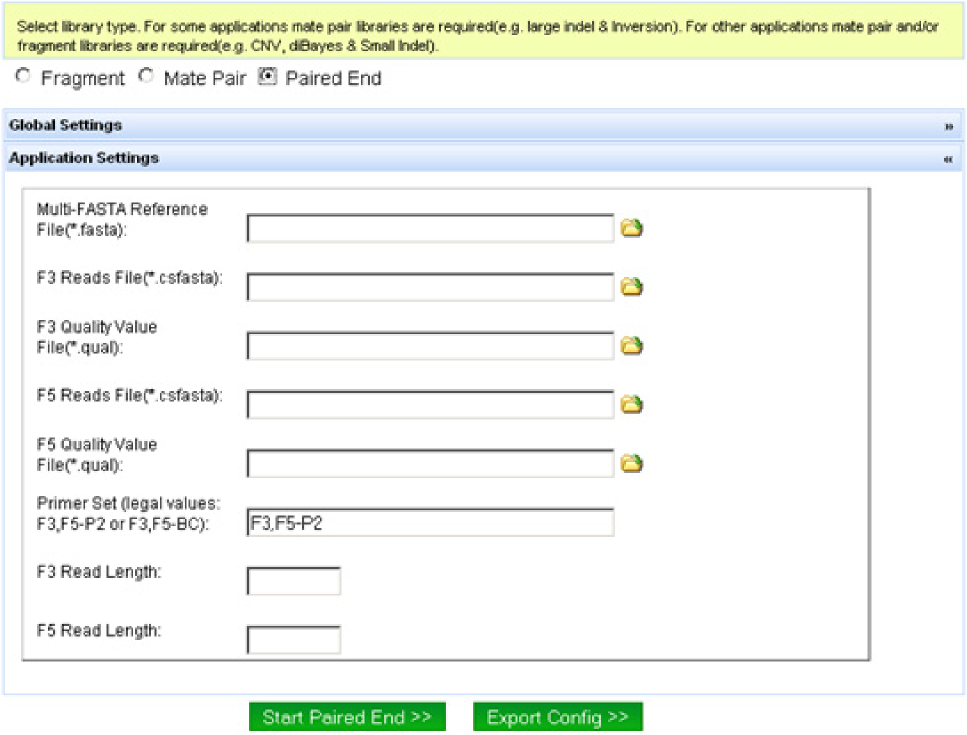
3. Define the directory path to the *.fasta file.
4. Click **Open**.
5. Click 📂 in the F3 Reads File(*.csfasta) field. The File Browser window appears.
6. Define the directory path to the *.csfasta file.
7. Click **Open**.
8. Click 📂 in the F3 Quality Value File (*.qual) field.
9. Define the directory path to the *.qual file.
10. Click **Open**.
11. Click 📂 in the R3 Reads File(*.csfasta) field.
12. Define the directory path to the R3 Reads File(*.csfasta) file.
13. Click **Open**.
14. Enter the read length in F3 Read Length.
15. Enter the read length in R3 Read Length.
16. Click Start Mate Pair >> to start the run.
17. At the job submission dialog, click **OK** after you have verified the folder locations.

See "Check the status of the run from the web interface" on page 134 to view the status of the mapping run.

Start the Map Paired-End data tool run

If you are mapping paired-end data, you must enter the F3 and R3 read lengths, define the primer set values, and define the absolute path to the following files (see Figure 37 on page 133):

- Multi-FASTA Reference File (*.fasta)
- F3 Reads File(*.csfasta)
- F3 Quality Value File (*.qual)
- F5 Reads File(*.csfasta)
- F5 Quality Value File(*qual)



Select library type. For some applications mate pair libraries are required(e.g. large indel & Inversion). For other applications mate pair and/or fragment libraries are required(e.g. CNV, diBayes & Small Indel).

○ Fragment ○ Mate Pair ● Paired End

Global Settings »

Application Settings «

Multi-FASTA Reference File(*.fasta):

F3 Reads File(*.csfasta):

F3 Quality Value File(*.qual):

F5 Reads File(*.csfasta):

F5 Quality Value File(*.qual):

Primer Set (legal values: F3,F5-P2 or F3,F5-BC): F3,F5-P2

F3 Read Length:

F5 Read Length:

Figure 37 Application Settings to map paired-end data

1. Click **Paired End**.
2. Click 📂 in Multi-FASTA Reference File(*.fasta). The File Browser window appears.
3. Define the directory path to the *.fasta file.
4. Click **Open**.
5. Click 📂 in F3 Reads File(*.csfasta). The File Browser window appears.
6. Define the directory path to the *.csfasta file.
7. Click **Open**.
8. Click 📂 in F3 Quality Value File (*.qual).
9. Define the directory path to the *.qual file.
10. Click **Open**.
11. Click 📂 in F5 Quality Value File (*.qual).
12. Define the directory path to the *.qual file.
13. Click **Open**.

14. Enter a value for the Primer Set.

15. Enter the read length in F3 Read Length.

16. Enter the read length in F5 Read Length.

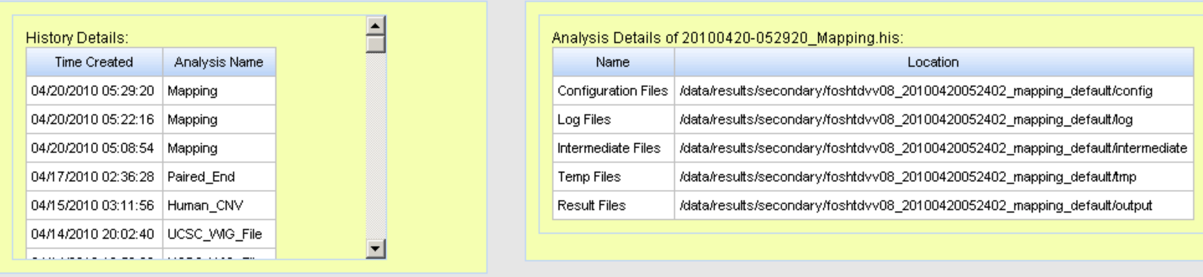
17. Click Start Paired End >> to start the run.

18. At the job submission dialog, click **OK** after you have verified the folder locations.

Check the status of the run from the web interface

1. Click **History**. The History window appears and the History Details table is displayed in the left pane. The History Details table shows the Time Created and Analysis Name for all runs performed on the BioScope™ Software cluster.

2. Scroll the History Details table and select a Mapping run, based on the data in the Time Created column (see Figure 38).



History Details:

Time Created	Analysis Name
04/20/2010 05:29:20	Mapping
04/20/2010 05:22:16	Mapping
04/20/2010 05:08:54	Mapping
04/17/2010 02:36:28	Paired_End
04/15/2010 03:11:56	Human_CNV
04/14/2010 20:02:40	UCSC_WIG_File

Analysis Details of 20100420-052920_Mapping.his:

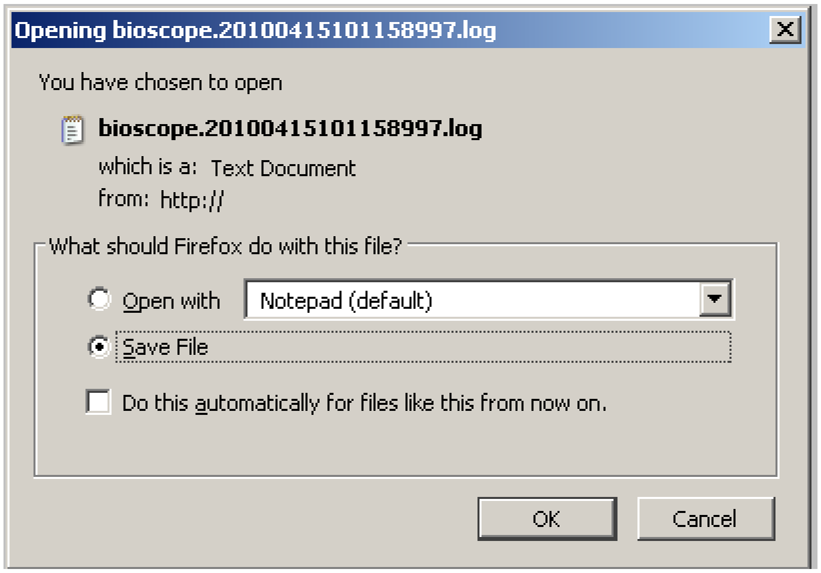
Name	Location
Configuration Files	/data/results/secondary/foshtdvv08_20100420052402_mapping_default/config
Log Files	/data/results/secondary/foshtdvv08_20100420052402_mapping_default/log
Intermediate Files	/data/results/secondary/foshtdvv08_20100420052402_mapping_default/intermediate
Temp Files	/data/results/secondary/foshtdvv08_20100420052402_mapping_default/tmp
Result Files	/data/results/secondary/foshtdvv08_20100420052402_mapping_default/output

Figure 38 History details and analysis details for a Map Data run

3. Double-click the Log Files row in the Analysis Details table. The File Browser dialog opens. Click **Resend** if your browser displays a message.

4. Select the `bioscope.`*`yyyymmddhhmmss`*`.log` file.

5. Click **Download**.
   - Click **Open with** and click **OK** to view the log file in Notepad or select a different text editor.
   - Click **Save File** to copy the file to your workstation.



Opening bioscope.20100415101158997.log

You have chosen to open

bioscope.20100415101158997.log

which is a: Text Document
from: http://

What should Firefox do with this file?

○ Open with Notepad (default)

● Save File

□ Do this automatically for files like this from now on.

Figure 39 Log file download page example

6. Scroll to the end of the file.

The run is complete if you see an entry similar to:

```
15 Apr 2010 03:16:32,537  INFO [main] PluginJobManager:130 -
>>>> END of PluginJobManager >>>> date DURATION=4 minutes 33
secs

15 Apr 2010 03:16:32,537  INFO [main] EventTransportFactory:129
- Closing JMS connection and session
```

Mapping results file formats

For information about the *.bam file that is generated by mapping, see Appendix A, “File Format Descriptions” on page 295.

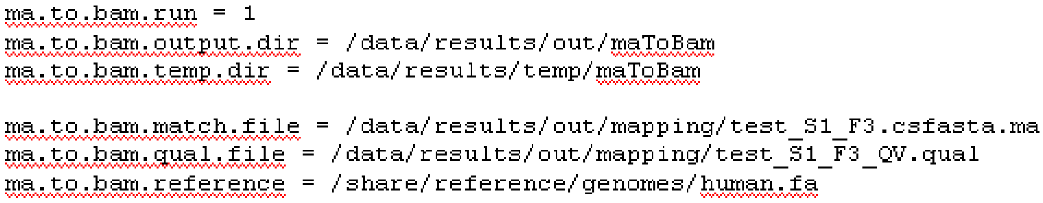
BAM file generation for fragment runs

A critical step in the resequencing pipeline is the generation of the BAM file. All tertiary analysis tools (for example, SNP calling) use a BAM file as input (see Appendix A, “Pairing information in a *.bam file” on page 301 for information on the BAM specification). In addition to converting mapping and pairing results to an industry standard format, the conversion process incorporates all of the logic to translate color space information to base space.

Single read data

Match results from fragment libraries are annotated and converted to BAM format using the MaToBam plugin. At a minimum, the MaToBam plugin takes a match file, a qual file, and a reference to generated the BAM file.

Figure 40 shows the parameters required for the MaToBam plugin.



```
ma.to.bam.run = 1
ma.to.bam.output.dir = /data/results/out/maToBam
ma.to.bam.temp.dir = /data/results/temp/maToBam

ma.to.bam.match.file = /data/results/out/mapping/test_S1_F3.csfasta.ma
ma.to.bam.qual.file = /data/results/out/mapping/test_S1_F3_QV.qual
ma.to.bam.reference = /share/reference/genomes/human.fa
```

Figure 40 Minimal MaToBam ini file example

Qual files are important for conversion to BAM format. Besides populating the color quality attribute, the contents of qual files are used to calculate base qualities that may be used in downstream applications. Prior to BioScope™ Software v1.2, many applications did not take advantage of color quality values and, as a result, these files may not have been saved by users with storage limitations. If this is the case, qual files can be simulated by writing a qual file with a defined value (for example, 30). If this is done, it is important that the entries be written in the same order as the input match file.

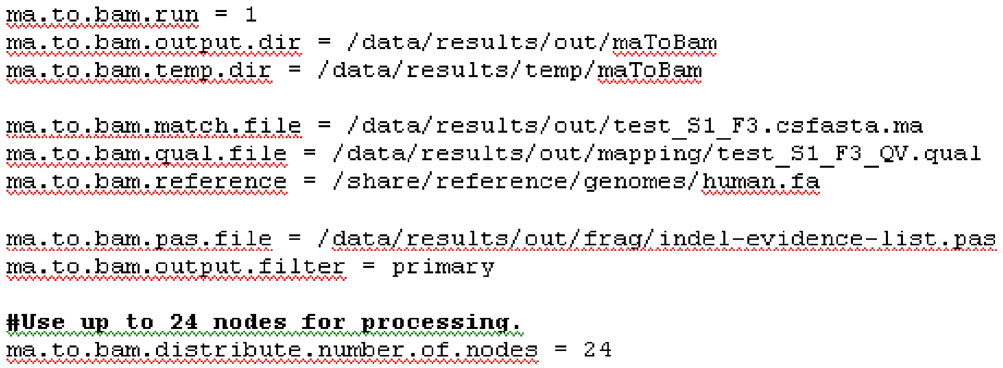
Selecting an appropriate output filter for MaToBam is important for the balance between the most complete dataset and the smallest possible disk footprint. In most cases, the default, `primary`, will be the correct option. One alignment is written for each bead, along with all gapped alignments if a Pas file is provided. This is similar to the `convert=beads` option from the legacy MaToGff tool and is appropriate for most fragment-based tertiary analysis, such as diBayes and the Small Indel tool. Users who need a complete, BAM-format record of the match file can use the `none` option to get a record for every alignment. It is important to note that this can be a very resource intensive option requiring large temporary space allocations and is best performed with a high level of parallelization.

Unique subsets can be generated for fragment BAM files in a couple of ways. Similar to the `convert=unique` option for MaToGff, the MaToBam plugin offers the `alignment_score` output filter, which can use a clear zone and mismatch penalty to report alignments for a bead whose score is much better than the next-best alignment. More commonly in the SAM/BAM community, however, is the use of the mapping quality score. The best quality score for a given application may differ and a series of values should be tried. Subsets can be made using the `samtools view` command as follows:

```
$ samtools view -q 20 /data/results/out/MaToBam/test.bam -o
/data/results/out/MaToBam/test.q20.bam
```

The MaToBam conversion can be a very resource-intensive process. Components of BAM files that exceed 100 Gb must be merged and sorted. To mitigate this, it is valuable to select a more restrictive output filter (for instance, `primary`) if possible. Additionally, MaToBam can be parallelized to a higher degree by increasing the value of the `ma.to.bam.distribute.number.of.nodes` parameter key.

Figure 41 shows an ini file for the MaToBam plugin, with the `ma.to.bam.distribute.number.of.nodes` parameter.



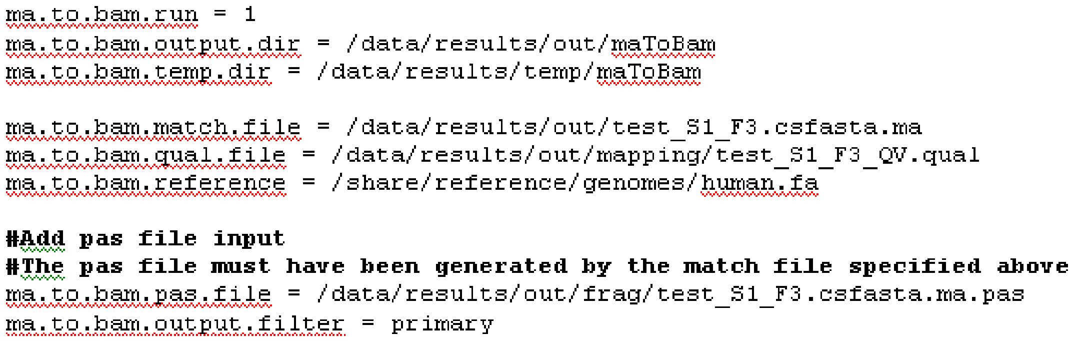
```
ma.to.bam.run = 1
ma.to.bam.output.dir = /data/results/out/maToBam
ma.to.bam.temp.dir = /data/results/temp/maToBam

ma.to.bam.match.file = /data/results/out/test_S1_F3.csfasta.ma
ma.to.bam.qual.file = /data/results/out/mapping/test_S1_F3_QV.qual
ma.to.bam.reference = /share/reference/genomes/human.fa

ma.to.bam.pas.file = /data/results/out/frag/indel-evidence-list.pas
ma.to.bam.output.filter = primary

#Use up to 24 nodes for processing.
ma.to.bam.distribute.number.of.nodes = 24
```

Figure 41 Example MaToBam ini file with an increased number of nodes

The Small Indel Tool, which calls indels from a consensus of gapped alignments, uses the BAM file as input. To incorporate the gapped alignments generated by the Frag Indel plugin, the Pas file should be added to the MaToBam inputs. In order to have the gap alignments be correctly associated with the non-indel ones in the BAM file, the PAS file must be produced with the same Match file input as the maToBam input here. (See "Determining gap alignments" on page 124 for more information on PAS file generation.)

The example MaToBam ini file in Figure 42 shows how to include Pas file input.

Note: `Primary` is the default value for the `ma.to.bam.output.filter` parameter. It is explicitly included in this ini example in order to indicate its importance.



```
ma.to.bam.run = 1
ma.to.bam.output.dir = /data/results/out/maToBam
ma.to.bam.temp.dir = /data/results/temp/maToBam

ma.to.bam.match.file = /data/results/out/test_S1_F3.csfasta.ma
ma.to.bam.qual.file = /data/results/out/mapping/test_S1_F3_QV.qual
ma.to.bam.reference = /share/reference/genomes/human.fa

#Add pas file input
#The pas file must have been generated by the match file specified above
ma.to.bam.pas.file = /data/results/out/frag/test_S1_F3.csfasta.ma.pas
ma.to.bam.output.filter = primary
```

Figure 42 MaToBam ini file example with Pas file input from the Frag Indel plugin

As in the MaToGff converter from previous SOLiD^{TM} software releases, MaToBam supports correction of color inconsistencies when converting to base space. If the value of the `ma.to.bam.correct.to` key is "reference", then the reference base will be used. If the value is "missing", an N is written. Correcting to "reference" is valuable for applications that do not process N very well, while correcting to "missing" helps avoid reference bias.

In some cases, you may want to process data that was paired through the MaToBam pipeline one tag at a time. Note that this will result in an LB field that indicates a fragment library (for example, 50F) since downstream tools will be unable to process the data as pairs.

An example complete MaToBam.ini file is shown below:

```
#####################################
```

```
# ma to bam pipeline parameters
####################################

# Parameter specifies whether to run maToBam plugin. [1 - run, 0
- do not run]
ma.to.bam.run = 1

# Parameter specifies the intermediate directory used by maToBam
plugin
# Default value when not specified is ${output.dir}/../
intermediate/maToBam
#ma.to.bam.intermediate.dir=${output.dir}/../intermediate/
maToBam

# Parameter specifies the temp directory used by maToBam plugin
# Default value when not specified is ${output.dir}/../temp/
maToBam
#ma.to.bam.temp.dir=${output.dir}/../temp/maToBam

# Parameter specifies the input pas file for maToBam
#ma.to.bam.pas.file=

# Parameter specifies the output filter to be used for
alignments. [primary|alignment_score|none]
# 'primary' - For each bead, output only the alignment with the
highest quality value.  Do output unmapped reads.

# 'alignment_score' - For each bead, output only the alignment
with the highest alignment score provided
# that the score exceeds the second highest score by the clear
zone. Do not output unmapped reads.

# 'none' - Output all alignments. Do output unmapped reads.

# Default value of the parameter when not specified is 'primary'
#ma.to.bam.output.filter=primary

# Parameter specifies which read-colors to be replaced
[reference|missing|singles|consistent]
# 'reference' - Replaces all read-colors annotated
inconsistent(i.e. 'a' or 'b') with the corresponding reference
color.

# 'missing' - Replaces all inconsistent read-colors with '.'.
These will translate to 'x' in the base space representation,
attribute 'b'.

# 'singles' - Replaces all 'single' inconsistent colors (i.e.
those annotated 'a' or 'b' and not adjacent to another 'b')
# with the corresponding reference color. Replaces all other
inconsistent colors with '.'.

# 'consistent' - For each block of contiguous inconsistent
colors, replaces the lowest QV - value color-call
```

```
# with the unique color that makes the block consistent. Breaks
ties at random.

# Default value of the parameter when not specified is
'reference'
#ma.to.bam.correct.to=reference

# Parameter is used in combination with conversion type set to
unique.
# Default value when not specified is 5
#ma.to.bam.clear.zone = 5

# Parameter is used to calculate the local alignment score with
conversion type set to unique.
# It must be a negative value. Ideally, it is the same value
used for MapReads.
# Default value when not specified is -2.0
#ma.to.bam.mismatch.penalty = -2.0

# Parameter specifies the library type. Currently 'fragment' is
the only option for this parameter.
# Default value when not specified is 'fragment'
#ma.to.bam.library.type=

# Parameter specifies the library name to be used.
# The value of the parameter along with library type is used to
create the LB field in BAM file
# Default value of the parameter when not specified is 'lib1'
#ma.to.bam.library.name=

# Parameter specifies the slide name. A typical value is of the
form liz_20091230_2.
# The value when specified is used in the PU header of the BAM
file
#ma.to.bam.slide.name=

# Parameter specifies the Ma To Bam Read Group description
#ma.to.bam.description=

# Parameter specifies the Read Group Sequencing center
# Default value of the parameter when not specified is
'freetext'
#ma.to.bam.sequencing.center=

# Parameter controls the type of variants reported. [a|ag|agy]
# 'a'   - Isolated single-color mismatches
# 'ag'  - Color position that is consistent with an isolated
one-base variant
# 'agy' - Color position that is consistent with an isolated
two-base variant

# Default value of the parameter when not specified is 'agy'
#ma.to.bam.tints=agy
```

```
# Parameter specifies the maximum base of the quality value
# Default value of the parameter when not specified is 40
#ma.to.bam.base.qv.max=40

# Parameter specifies the output folder where mapping results
reside
# If the parameter ma.to.bam.match.file is not specified then
all the .ma files in the older
# are converted to BAM files.
# This parameter need not be set if a full path to the file is
provided by ma.to.bam.match.file parameter
mapping.output.dir = ${output.dir}/s_mapping

# Parameter specifies the mapping output file to be used for
conversion.
# User can provide the full path or use it in combination with
mapping.output.dir mentioning just the file name
#ma.to.bam.match.file=${output.dir}/s_mapping/
Rosalind_20080729_2_Chris5_F3.csfasta.ma.50.2

# Parameter specifies the quality file to be used for conversion
# User can provide the full path of the quality file including
the file name or mention the file name and use it combination
with mapping.output.dir
ma.to.bam.qual.file = ${output.dir}/s_mapping/Rosalind
```

Table 21 describes the parameters used with MaToBam.

Table 21 Parameters for MaToBam runs

Parameter name	Default value	Description
Input/output parameters		
ma.to.bam.output.dir, ma.to.bam.output.file		Use of output dir should get you a stereotyped file name for the mapped and unmapped BAM file. If an output.file is specified and an unmapped bam file is produced, it will have the same name as output.file, except for "unmapped.bam" instead of "bam" extension.
ma.to.bam.temp.dir		Temp dir
ma.to.bam.match.file		Match file (required)
ma.to.bam.qual.file		qv file (required)
ma.to.bam.pas.file		Pas file
reference		Reference (required)
Annotation parameters		

Parameter name	Default value	Description
ma.to.bam.output.filter	primary	Subset of data to be written. Value can be primary, alignment_score or none. • primary: Reports only the primary alignment. Each bead id has a single primary alignment that corresponds to the highest mapping quality value. If multiple alignments have the same highest value one is randomly selected. Unmapped reads are placed in a separate file. All indel alignments are included. • alignment_score: – For reads with a single hit, a single corresponding BAM entry with mapping quality is reported. – For reads that have more than one hit, let s1 be the hits’ highest local alignment score, and s2 be their second highest. Then alignment1 is deemed to map uniquely if s1 –s2 > cz, where cz is the clear zone defined by the ma.to.bam.clear.zone option. If it is unique, alignment1 is reported in the BAM file with positive mapping quality. – For reads that have more than one hit that are not unique by the clear zone definition, the highest scoring alignment is reported, with mapping quality set to zero. • none: All alignments are reported.
ma.to.bam.clear.zone	5	The requested "clear zone". See ma.to.bam.output.filter=alignment_score.
ma.to.bam.mismatch.penalty	-2.0	The mismatch penalty used to calculate the local alignment score (see ma.to.bam.output.filter above) in the definition of ma.to.bam.output.filter= alignment_score. It must be a negative value. Ideally, it is the same value used for MapReads.
ma.to.bam.correct.to	reference	Specifies how to correct the color calls. Value can be reference, missing, singles, or consistent” • reference: Replaces all read-colors annotated inconsistent (i.e. 'a' or 'b') with the corresponding reference color. • missing: Replaces all inconsistent read-colors with '.'. These will translate to 'n' in the base space representation, attribute 'b'. • singles: Replaces all 'single' inconsistent colors (i.e. those annotated 'a' or 'b' and not adjacent to another 'b') with the corresponding reference color. Replaces all other inconsistent colors with '.'. • consistent: For each block of contiguous inconsistent colors, replaces the lowest QV-value color-call with the unique color that makes the block consistent. Breaks ties at random.
ma.to.bam.base.qv.max	40	Maximum value for base qv.
ma.to.bam.tints	agy	Controls the type of variants reported. The values are: • a: Isolated single-color mismatches • ag: Color position that is consistent with an isolated one-base variant • agy: Color position that is consistent with an isolated two-base variant
Read group information		

Parameter name	Default value	Description
ma.to.bam.library.type		Must be "fragment".
ma.to.bam.slide.name		Typical slide name (for example, liz_20091230_2). Goes into "PU" header.
ma.to.bam.library.name		Free text. Concatenated with library type to populate the LB: field.
ma.to.bam.description		Free text. Read group description (DS)
ma.to.bam.sequencing.center		Free text. Read group sequencing center (CN)
Global read group information		
sample.name		Free text. Read group sample name (SM)

Prepare to run the Map Data tool

Run the MaToBam tool on the command-line

Although several different software programs are involved in the experiment, a single command generates all of the related programs required to complete the experiment. The *.plan file that is specified in the command syntax controls the order in which the BioScope™ Software runs the related programs.

Start a MaToBam run

1. Connect to the BioScope™ Software cluster and login with a user ID that has write privileges on all of the directories that BioScope™ Software uses when the tool runs.

2. Create the MaToBam ini file, as described in "BAM file generation for fragment runs" on page 135.

3. At a command prompt, enter:

   ```
   bioscope.sh -l filename.log filename.plan
   ```

   Do not log out of the BioScope™ Software cluster.

To check the run status from the command line:

1. Navigate to the log directory that is defined in the mapping.ini file. For example, you might enter:

   ```
   cd /data/results/tertiary/mapping/log
   ```

2. Open `bioscope.`*yyyymmddhhmmss*`.log`, where *yyyymmddhhmmss* is the timestamp when the file is created.

3. Scroll to the end of the file.

4. The run is complete if you see an entry similar to:

   ```
   20 March 2010 12:37:58,304 INFO [main] PluginJobManager:104
   - >>>> END of PluginJobManager >>>> date DURATION=949
   millisecs.
   ```

FAQs – Mapping

1

How does the local alignment approach affect mapping?

Using the local alignment approach removes the constraint of a whole-read alignment, and mapping rate is significantly increased. It is not uncommon for a poor dataset with 30% mapping rate at 50_6 using previous versions of BioScope™ Software to reach more than a 60% mapping rate with the mapping algorithm that is in BioScope™ Software v1.2 and later releases.

While the majority of alignments are full length, some alignments can vary in length. If a read has many errors towards the end, the final alignment will not include these positions if a shorter alignment receives a better overall score.

2

How do I specify seeds for local alignments?

A seed can be specified by three parameters:

- The seed length.
- The quantity of allowed mismatches in the seed.
- The start site of the seed within the read.

For example, you might look at the first 25 bases of a 50 bp reads, and attempt seed extension if that portion aligns to the reference with two or fewer mismatches (see Figure 43.)

A seed like the one mentioned in the example would be abbreviated 25.2.0, which means that the seed is 25 bases long, allows two mismatches, and starts at the beginning of the read.

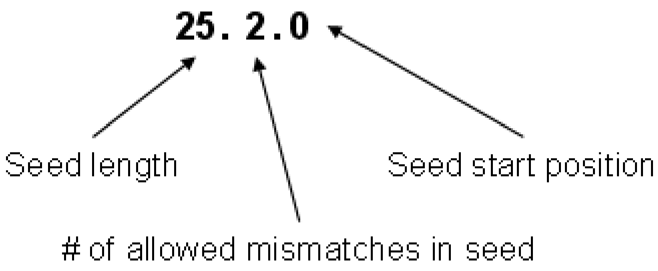


Figure 43 Seeds for local alignments

3

How do I increase mapping rate?

Mapping rate increases with each additional round of mapping. However, the gain in rate comes at the cost of increased runtime and disk requirement. Testing has found sets of mapping schemes that strike a good balance between mapping rate and runtime for 25-, 35,- and 50-bp reads. These schemes are shipped as BioScope™ Software defaults and should work well for most purposes.

For 50-bp reads, two keys in the mapping.ini file define how many rounds of mapping are run, and what happens in each round. The default values of the keys are shown below, and their meanings are further explained in Table 19 on page 121.

Note: Repetitive schemes are empty by default.

- mapping.scheme.unmapped.50 = 25.2.0,25.2.15
- mapping.scheme.repetitive.50 = 38.3.0,25.2.0

4

What is a good seed for my application?

Consider the following factors when picking seed parameters:

- Mapping is slower when more mismatches are allowed in the seed. However, allowing more mismatches improves the mapping rate.
- Shorter seeds have higher mapping rates, but also lead to more spurious alignments.
- Color-calls at the beginning of the read are more reliable than those at the end. Therefore, you have the option to anchor the seed near the front of the read. However, applications such as transcriptome sequencing are scenarios where it is an advantage to anchor the seed near the end of the read.

For 50-bp reads, the default setting of 25.2.0 for the first round, followed by 25.2.15 in the second round, delivers good results in most cases. For a single round, 30.3.0 is recommended.

5

What happens when I run multiple rounds of mapping?

The parameters for the example shown in Figure 44 are set in the mapping.ini as follows:

- mapping.scheme.unmapped.50 = 25.2.0,25.2.15
- mapping.scheme.repetitive.50 = 38.3.0,25.2.0

In this case, four rounds of mapping are specified. Each read will go through the following decision tree:

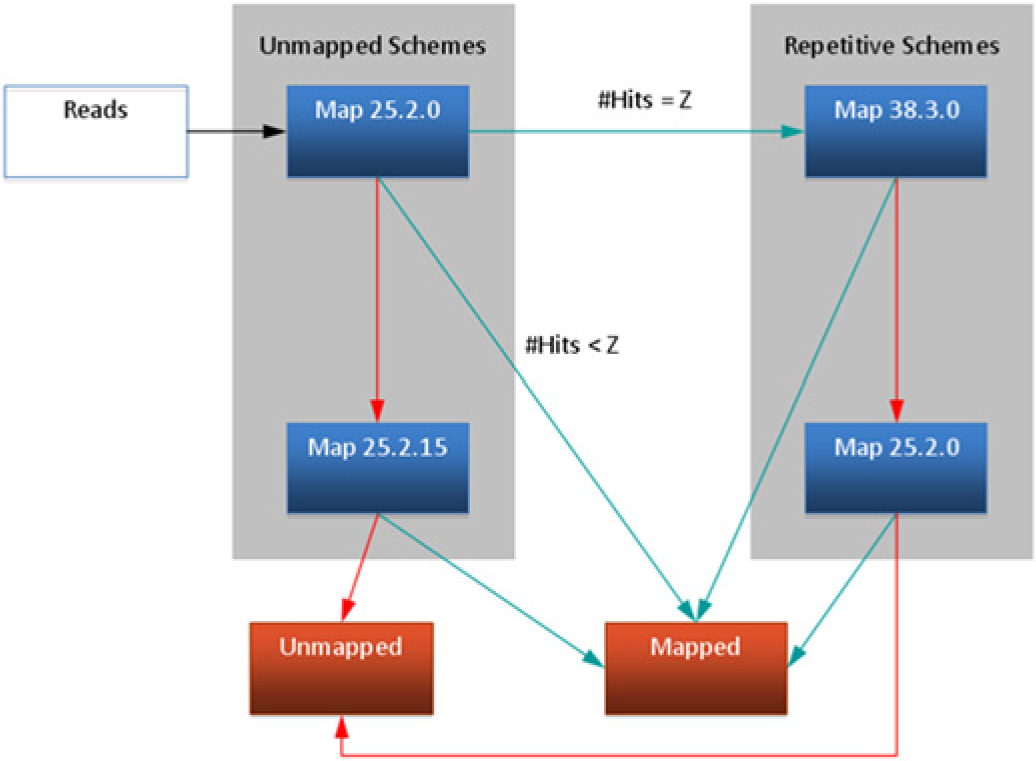


Figure 44 Read decision tree

The following paragraphs refer to Figure 44.

After every round, aligned reads follow the blue arrow. Unaligned reads follow the red arrow. When a read is considered to be mapped, it does not participate in further rounds of mapping.

Because there are two blue arrows depending on the number of alignments that the read has, the first mapping round of the unmapped schemes is considered to be a special case. If the number of alignments reaches the *Z* threshold set by matching.max.hits, then the read would go into the repetitive schemes. If the number of hits is fewer than *Z*, then the read is considered mapped.

The 25.2.0 round specified at the end of the repetitive schemes might seem redundant, considering that all reads initially go through a 25.2.0 round. The reasoning is that if a read has *Z* or more alignments with 25.2.0, it might map to fewer locations with a more restrictive seed such as 38.3.0. For those reads that fail to map with 38.3.0, the 25.2.0 round is rerun to recover the *Z* alignments. Only a small subset of reads is expected to reach this point, and the final round of 25.2.0 takes only a fraction of the time it takes to run the first 25.2.0 round.

6

How do I interpret mapping quality value?

First of all, it is worth noting that mapping quality value (QV) is not like the p-value in BLAST. In BLAST, the p-value is used to estimate the likelihood that the aligned sequences are related. In general, reads come from the same source and it is known that the two are related.

The purpose of mapping QV is to estimate the probability that the read originates from the mapped genomic location. The estimate is determined mainly by the difference in significance between the best- and second-best hits.

The numeric interpretation of mapping QV is the same as the base call QV. A mapping QV of ten means that there is a 90% chance that the alignment is correct. A mapping QV of 20 means that there is a 99% chance that the alignment is correct.

7

How much RAM do I need to analyze human samples?

For optimal performance, 24 Gb of RAM per cluster node is recommended for human samples. If the cluster node has less than 24 Gb of available RAM, mapreads can split the genome into smaller segments, and align to each segment sequentially. Splitting the genome into smaller segments is implemented inside mapreads and is completely transparent to the user.

Mapping with less than the recommended amount of RAM slows down performance. If the cluster node has 16 GB of RAM, mapping human samples will be roughly 30% slower compared to a machine with 24 GB of RAM.

8

What should I change in the ini file from run to run when running from the command line?

Below is a sample mapping.ini file for 50 base-pair reads. The parameter that is highlighted in **bold** must be changed for every mapping run. Those that are highlighted in *italics* need to be verified to make sure they are appropriate for the run. Finally, the grey lines can remain as they are in most cases.

Note: You can use the variable expansion in the mapping.ini file. For example, by defining output.dir and tmp.dir relative to base.dir, it is possible to update multiple keys just by changing the value assigned to base.dir.

```
pipeline.cleanup.middle.files=1
pipeline.cleanup.temp.files=1

#    Global Parameters
primer.set=F3
run.name=Run1
sample.name=Huref_frag
read.length=50
base.dir=/data/results/secondary/siena_20091020_1
```

```
reference=/share/reference/genomes/hg18Validated/
hg18Validated.fasta
output.dir=${base.dir}/output
tmp.dir=${base.dir}/tmp
intermediate.dir=${base.dir}/intermediate
log.dir=${base.dir}/log
scratch.dir=/scratch/solid

######################################
##  mapping.run
######################################
mapping.run=1
mapping.tagfiles.dir=${output.dir}/qvfiltered
mapping.output.dir=${output.dir}/s_mapping
mapping.run.classic=false
mismatch.level=2
matching.max.hits=100
mapping.mismatch.penalty=-2.0
mapping.qual.filter.cutoff=0
mapping.scratch.dir=/scratch/solid
mapping.scheme.unmapped.50=25.2.0,25.2.15
mapping.scheme.repetitive.50=
```

10 Run the Resequencing Pairing Tool

This chapter covers:

Pairing algorithm description

The BioScope™ Software pairing pipeline supports both mate-pair and paired-end pairing experiments.

Mate-pair algorithm

The pairing tool matches pairs of reads from the F3 and R3 mapping results of a mate-pair run. Pairing also matches reads from the F3 and F5-P2 files of a paired-end mapping run.

The tool also enables mate-pair rescue. Mate-pair rescue is an additional matching process that uses information from the library preparation in a mate-pair sample. The pairing tool creates reports about mate-pair quality and pairing statistics.

The pairing algorithm performs the following steps for each pair of reads:

1. It finds all "good AAA" pairs based on the order, orientation, and distance between the two reads.

   Note: Refer to "FAQs – Pairing" on page 284 for information about, and definitions of, three-letter genomic code classifications.

2. If no AAA pair is found, and a reference sequence is available, the algorithm performs mate-pair rescue. Mate-pair rescue is accomplished by using hits to one tag as an anchor, and then scanning for the "sister tag" in the region predicted by the library insert size.

   Note: Either F3 or R3 tags can serve as the anchor, as long as the number of hits is below the Z threshold. The Z threshold is determined by the value entered for the mapping.max.hits parameter. If the F3 tag has x alignments, and the R3 tag has y alignments, and both tags have less than Z hits, then the x+y anchor candidates are examined (see Figure 45).

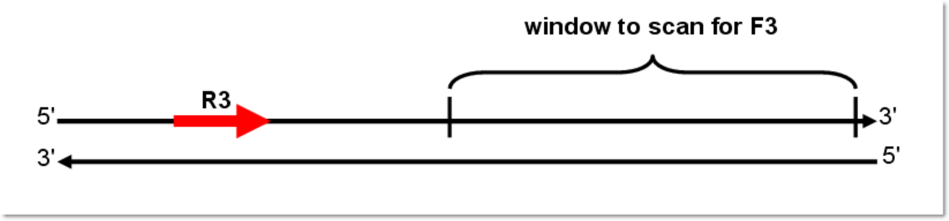


Figure 45 Mate-pair rescue example

3. It determines if a AAA pair is unique. If multiple AAA pairing candidates exist for a given pair of tags, then the pair that scores at least x higher than the second-highest scoring pair is still considered unique, and all other AAA candidates with lower scores are discarded. The value of x is determined by the setting of the pair.uniqueness.threshold parameter in the pairing.ini file.

4. For non-AAA pairs where both tags have unique hits, the algorithm performs additional classifications based on the strand, distance, and orientation of the tags.

If either of the two tags has multiple mapping locations, but the highest mapping score is more than half of x higher than the second-highest score, then the location with the highest score is still considered unique and all other locations are discarded.

The pairing tool supports paired-end experiments in addition to mate-pair experiments. In paired-end experiments, the actual sequencing is done on the same strand in opposite directions. However, because of the representation in the csfasta file, the matching and pairing pipelines attempt to match the F5 and F3 tags on different strands, facing each other. In addition, a distance constraint determined by insert size is satisfied (see Figure 46). The pairing algorithm follows the same steps as the mate-pair algorithm. Steps 1 and 2 are modified to enforce the different order and orientation requirement. If you convert matching positions for F3 tags to the opposite strand, then paired-end pairing is performed the same way as it is with mate-pair. The three-letter classification is defined the same way in mate-pair and pair-end. (See Table 24 and Table 25 on page 168 for the genomic classification tables.)

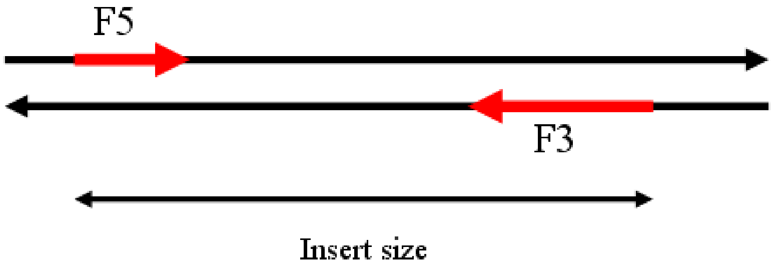


Figure 46 Paired-end tags example

Paired-end algorithm

The mapping reads generated from the paired-end sequencing runs are similar to the mapping mate-pair reads. The combination of these two primer pairs provides a way to identify structural differences and rearrangements in whole genomes and whole transcriptomes. In paired-end ligation, the F3 (forward) primer binds to the P1 adapter, while the F5 (reverse) primer binds to the P2 adapter. Although the DNA insert fragment between the P1 and P2 adapters has a variable length, the F3 read length is 50 bp and the F5 read length is 25 bp upon release.

Some individual BioScope™ Software analysis tools support paired-end data (see Table 5 on page 46). The RNA splicing and gene fusion detection applications use paired-end input. For WTA with a paired-end RNA library, the sequence reads are mapped with BioScope™ Software. You can use the resulting data to detect splice junctions and gene fusions.

Algorithm for calculating Pairing Quality Values

The pairing algorithm reports multiple sets of possible alignments for any given pair of reads (F3/R3 tags for a mate-pair run and F3/F5-P2 tags for a paired-end run). The pairing quality algorithm uses a Bayesian approach to calculate the quality of a given alignment for a pair of reads and the alignment with the highest pairing quality value (PQV) is chosen as the primary alignment for the pair of reads. The PQVs represent the Phred-scaled quality score, and are useful for downstream variant detection tools such as DiBayes, small indels, large indels, and CNV.

Quality of any given alignment for a pair of reads r_1, r_2 mapped to positions x_1 and x_2 in the reference genome is represented by:

$$Q(r_1, r_2, x_1, x_2) = P(A(r_1, r_2, x_1, x_2)|r_1, r_2)$$

where A represents the event when reads r_1, r_2 are sequenced from locations x_1 and x_2 respectively and $P(A|\ r_1, r_2)$ is the probability of the event A occurring given the pair of reads r_1 and r_2.

Using the Bayesian approach, the posterior probability $P(A|\ r_1, r_2)$ is given by:

$$P(A(r_1, r_2, x_1, x_2)|r_1, r_2) = \frac{P(r_1,r_2|A) \times P(A)}{P(r_1, r_2)}$$

The probability $P(r_1,r_2)$, of finding reads r_1 and r_2 is a function of the complexity of the genome sequenced. For the purpose of simplicity we calculate the probability as:

$$P(r_1, r_2) = \sum_{i,\ j \in M} P(r_1, r_2|A(r_1, r_2, i, j)) \times P(A(r_1, r_2, i, j))$$

where M is the set of all possible alignments to the reference genome for reads r_1 and r_2. Using this relationship to represent $P(\ r_1, r_2)$ in the previous equation we get:

$$P(A(r_1, r_2, x_1, x_2)|r_1, r_2) = \frac{P(r_1, r_2|A(r_1, r_2, x_1, x_2)) \times P(A(r_1, r_2, x_1, x_2))}{\sum_{i,\ j} P(r_1, r_2|A(r_1, r_2, i, j)) \times P(A(r_1, r_2, i, j))}$$

The prior probability P(A) of the event A is further given by:

$$P(A(r_1, r_2, x_1, x_2)) = P(A(r_2, x_2)|B) \times P(B(r_1, x_1))$$

where $B(r_1,x_1)$ is the event that read r_1 is sequenced from location x_1 in the genome and $P(A|B)$ is the conditional probability of finding the event A where read r_2 is sequenced from location x_2, given that read r_1 was sequenced from location x_1.

The probability $P(B)$ is a constant for any given read r_1, and the conditional probability $P(A|B)$ should theoretically follow the insert-size distribution. For the sake of simplicity the following priors are assumed in the pairing quality calculations (see "FAQs – Pairing" on page 284 for definition of three letter genomic codes):

- P(A|B) = 1, for all 'AAA' pairs
- P(A|B) = 1/10,000, for all 'non-AAA' pairs (including small and large indels)
- P(A|B) = 1/10,000, when one of the reads in the pair cannot be mapped to the reference genome.

In cases where a pair of reads have a unique set of alignments to the reference genome, the posterior probability $P(A|\ r_1, r_2)$ would always result in 1, thereby obscuring the relative quality of the alignment compared to those of other read pairs. To overcome this issue, we calculate a background probability P_B which represents the probability of finding an alignment to the reference genome with M+1 mismatches, where M is the maximum allowed mismatches set in the pairing.ini file (with the `matching.max.hits` parameter). The formula for background probability P_B is:

$$P_B = P(r_1|A(r_1, x_1)) \times P(r_2|B, M + 1 \ mismatches),\ r_1 > r_2(k_1 > k_2,\ if\ r_1 = r_2)$$

In case of uniquely paired reads the posterior probability is given by:

$$P(A(r_1, r_2, x_1, x_2)|r_1, r_2) = \frac{P(r_1, r_2)|A}{P(r_1, r_2)|A + P_B}$$

For mapping using local alignment method, the likelihood function $P(r_1,r_2 \mid A)$ is given by:

$$P(r_1, r_2)|A = (1 - e)^{(k_1 + k_2) - (m_1 + m_2)} \times e^{(m_1 + m_2)} \times \frac{1}{4}^{(L_1 + L_2) - (m_1 + m_2)}$$

where L_1 and L_2 are the read lengths for reads r_1 and r_2 respectively, (for example, F3 = 50 and R3 = 50),
k_1 and k_2 are the alignment lengths ($k_1 \leq L_1$ and $k_2 \leq L_2$),
m_1 and m_2 are the number of mismatches, and
e is the error rate.

In order to be consistent with the Phred quality score (-10*log10[prob(error)]) used widely in literature, the PQV is computed as the negative log odds of misaligning the pair of reads:

$$\text{PQV} = -10 \times \log_{10} [1 - Q(r_1, r_2, x_1, x_2)]$$

The resulting pairing quality values are normalized by the maximum possible value to ensure that the pairing quality values are within the range [0,100].

$$\text{PQV} = \frac{\text{PQV}}{\text{PQV}_{\text{max}}} \times 100$$

PQV_{max} is the maximum possible pairing quality value when the pair of reads map uniquely to the reference with zero mismatches.

Calculating PQVs for gapped alignments

The pairing algorithm searches for gapped alignments (indels) when one of the tags (F3/R3/F5-P2) maps to the reference genome and the other tag does not map to the genome within the insert-size range. If both an ungapped and a gapped alignment are found for a given read, then, due the low prior probability of 10^-4 assigned to the gapped alignments, the PQV for gapped alignments will be zero.

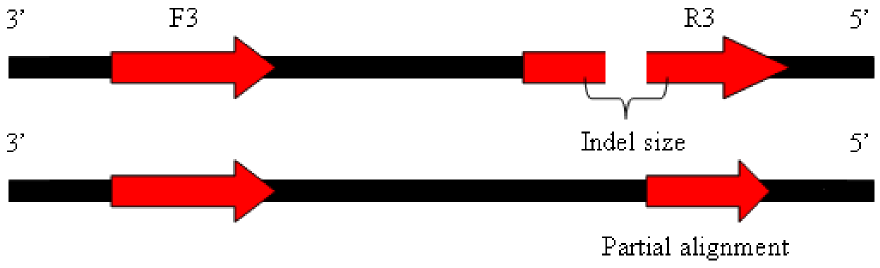


Figure 47 Example of a gapped alignment and a partial alignment

In calculating the PQV for gapped alignments, the alternative hypothesis tested is the probability of finding the partial ungapped alignments. The read with the gapped alignment is treated as two partial reads on either side of the indel start point. The partial read with the greater length is used as the partial alignment length for the alternate hypothesis. This ensures that gapped alignments with an indel starting point at the middle of the read, and with significant length of alignment on either side of the indel starting point, are assigned a higher PQV compared to gapped alignments with an indel starting point close to either ends of the read.

$$P(A|r)_{Indel} = \frac{P(r|A)_{Indel}}{P(r|A)_{Indel} + P_{PartialAlignment}}$$

Assigning Primary Alignment

For reads with multiple ungapped alignments, the one with the highest PQV is chosen as the primary alignment for the read and is reported to the BAM file. In cases where there are multiple alignments with the same PQV, then the primary alignment is chosen at random from among the alignments with the same PQV.

Mate-pair pairing.ini file example

The following section shows a typical example of the pairing.ini file. See Table 22 on page 157 for a description of the pairing.ini file parameters.

IMPORTANT! Before you begin a run, you must verify the settings for each parameter highlighted in **bold** in the *.ini file example shown in the next section.

```
# To include some common variables.
import ../globals/global.ini
#Reference genome file name.
reference = ${reference.dir}/
DH10B_WithDup_FinalEdit_validated.fasta
reads.result.dir.1 = ${base.dir}/F3/reads1
reads.result.dir.2 = ${base.dir}/R3/reads2

##
***************************************************************
##   pairing
## ***************************************************************

# mandatory parameters
# --------------------
# Parameter specifies whether to run or not pairing pipeline.
[1: to run, 0:to not run]
pairing.run = 1
# Mapping output directories
mate.pairs.tagfile.dirs = ${base.dir}/F3/outputs/
s_mapping,${base.dir}/R3/outputs/s_mapping

pairing.output.dir = ${output.dir}/pairing

# optional parameters
# -------------------
# Selects a set of parameters for indel search:
# 1: Deletions to 11, insertions to 3, Small indels.
# 2. Deletions to Small indels. (not used)
# 3: Insertions from 4 to 14
# 4: Insertions from 15 to 20.
# 5: Longer deletions from 12 to 500.
# Any of the values 1, 3, 4, or 5 may be entered, separated by
comments
#indel.preset.parameters = 1,3,4,5

# Max Base QV. - The maximum value for a base quality value
#max.base.qv = 40

# Minimum Insert - Minimum insert size defining a good mate. If
this is not set the code will attempt to measure the best value
#insert.start =

# Maximum Insert - Maximum insert size defining a good mate. If
this is not set the code will attempt to measure the best value
#insert.end =

# Rescue Level - "Usually 2 * the mismatch level
#mate.pairs.rescue.level = 4

# Pairing statistics file name
#mates.stats.report.name = pairingStats.stats

# Max Hits for Indel Search
#indel.max.hits = 10

# Maximum Hits
#matching.max.hits = 100

# Mapping Mismatch Penalty
#mapping.mismatch.penalty = -2.0

# Parameter specifies the alignment size of the anchor region
#pairing.anchor.length=25

# Minimum Non-mapped Length for Indels
#indel.min.non-matched.length = 10

# Rescue Level for Indels - Default for 50mers, 3 for 35mers,
and 2 for 25mers.
#indel.max.mismatches = 5

# Use template Rescue File For Indels
#use.template.rescue.file = true

# Max mismatches in indel search for tag 1
#pairing.indel.max.mismatch.tag1 = 5

# Max mismatches in indel search for tag 2
```

```
#pairing.indel.max.mismatch.tag2 = 5

# Pair Uniqueness Threshold
#pair.uniqueness.threshold = 10.0

# Maximum estimated insert size
#max.insert.estimate = 20000

# Minimum estimated insert size
#min.insert.estimate = 0

# Primer set - Use this only when both directories specified by
mate.pairs.tagfile.dirs are the same. Then the files must have
these strings
# immediatly before the .csfasta, if present, or the .ma
extension.
# primer.set = F3,R3

# Mark PCR and optical duplicates
#pairing.mark.duplicates = true

# Color quality file path 1 - Color quality file path for first
tag. Use instead of reads directories.
#pairing.color.qual.file.path.1 =

# Color quality file path 2 - If either file path is explicitly
set, both must be.
#pairing.color.qual.file.path.2 =

# Annotations: How to correct color calls -
#Specifies how to correct the color calls.
# 'missing' - Replaces all inconsistent read-colors with '.'.
These will translate to 'x' in the base space representation,
attribute 'b'.
# 'reference' - Replaces all read-colors annotated inconsistent
(i.e., 'a' or 'b') with the corresponding reference color.
# 'singles' - Replaces all 'single' inconsistent colors (i.e.,
those annotated 'a' or 'b' and not adjacent to another 'b') with
the corresponding
#   reference color. Replaces all other inconsistent colors with
'.'.
# 'consistent' - For each block of contiguous inconsistent
colors, replace all single insistent colors
#   (i.e., those annotated 'a' or 'b' and not adjacent to another
'b') with the corresponding reference color. Replace all other
inconsistent
# colors with '.'.
# 'qvThreshold' - A scheme combining the four above choices,
based on the specified qvThreshold. (--correctTo: default is
missing)
#pairing.correct.to = reference

# Single-tint annotation - Represents any number of single-tint
annotations.
```

```
# 'a' - Isolated single-color mismatches (grAy).
# 'g' - Color position that is consistent with an isolated one-
base variant (e.g., SNP).
# 'y' - Color position that is consistent with an isolated two-
base variant.
#    (default is agy if not specified.)
#pairing.tints = agy

# User Library prefix - Prefix for LB attribute of BAM file.
Accepts any characters except tab and hyphen
#pairing.library.name =

# Parameter specifies the path to the ma result file of F3
mapping
#pairing.first.mapping.file=

# Parameter specifies the path to the ma result file of R3
mapping
#pairing.second.mapping.file=

# Parameter specifies filter for the records in the output  bam
file ['primary' specifies only primary alignments, 'none']
# Default value when not specified is 'primary'
#pairing.output.filter=primary

#################################
#################################
##
##  temp files and folders keep
##  # don't keep temp files and folders for clean run ...   make
it =1, if you want them to be deleted.
#job.cleanup.temp.files = 0
```

Mate-pair pairing.ini file parameter descriptions

Table 22 pairing.ini file parameter description

Parameter name	Default value	Description
pairing.run	—	Determines whether or not to run the pairing analysis. Allowed values are: • 0: Do not run the analysis. • 1: Run the analysis.
indel.preset.parameters	1,3,4,5	Selects a set of parameters for indel search: • 1: Searches for deletions to 11, insertions to 3. • 2: Not used. • 3: Search for insertions from 4 to 14. • 4: Search for insertions from 15 to 20. • 5: Longer deletions from 12 to 500.

Table 22 pairing.ini file parameter description *(continued)*

Parameter name	Default value	Description
insert.start	—	The minimum insert size to define a good mate. If a value is not set, the tool tries to measure the best value.
insert.end	—	The maximum insert size used to define a good mate. If a value is not set, the tool tries to measure the best value.
mate.pairs.rescue.level	4	The maximum mismatches allowed during rescue. The value is usually twice the matching mismatch level of the anchor. A value of zero indicates no rescue.
mates.stats.report.name	pairingStats.stats	The pairing statistics output file name generated in the folder given by `pairing.output.dir`.
reads.result.dir.1	—	The F3 reads directory.
reads.result.dir.2	—	The R3 reads directory.
mapping.mismatch.penalty	-2.0	The penalty to the mapping quality for a mismatch.
pairing.anchor.length	25	The alignment size of the anchor region.
indel.min.non-matched.length	10	The minimum non-mapped length for indels.
indel.max.mismatches	5	The maximum mismatches for indels. Use 2 for 2x25mer reads. Use 3 for 2x35mer reads.
use.template.rescue.file	1	Causes the indel search run of pairing to refer to the pairing run output file to avoid searching for pairs of reads that already have no results.
pairing.indel.max.mismatch.tag1	5	Maximum number of mismatches allowed for indel/ gap alignments on the F3 tag.
pairing.indel.max.mismatch.tag2	5	Maximum number of mismatches allowed for indel/ gap alignments on the R3 tag.
pair.uniqueness.threshold	10.0	If the best pair found has a quality value at least this many times larger than that of the second-best pair, accept the best pair as unique.
run.name	—	Used to annotate the indel BAM file.
sample.name	—	Used to annotate the indel BAM file.
mate.pairs.tagfile.dirs	${output.dir}/${primer.set.1}/s_mapping,${output.dir}/${primer.set.2}/s_mapping	A comma-separated pair of directory names that specify the location of the F3 and R3 mapping files. The tag file names are found by searching the directories. The files must end in *.ma. As an option, the *.ma file name can be followed by two numbers separated by dots. The two directories can optionally specify the same directory, provided the file names contain the primer set strings (F3 or R3) somewhere in the string to the left of the .ma suffix and the primer.set parameter is set to F3,R3.
primer.set	F3,R3	This parameter only needs to be set if the mate.pairs.tagfile.dirs value contains the same directory on both sides of the comma.

Table 22 pairing.ini file parameter description *(continued)*

Parameter name	**Default value**	**Description**
pairing.first.mapping.file	—	The complete file name of the F3 tag file. The tool only uses this parameter if the directories are not specified.
pairing.second.mapping.file	—	The complete file name of the R3 tag file. The tool only uses this parameter if the directories are not specified.
mapping.output.dir	—	This parameter key is present to establish dependency when this tool is called for in the same configuration file as the mapping tool.
pairing.output.dir	${output.dir}/pairing	The directory where the final output from pairing will be written.
indel.max.hits	10	The maximum number of hits allowed in both tags in an indel finding. The pairing tool stops looking for hits after it has found the specified number of hits for a bead (read-pair).
pairing.maximum.workers	24	The default number of pairing jobs to be run (provided nodes are available).
memory.requested	4 Gb	The amount of memory a cluster manager must allow for the jobs that run the pairing program.
max.insert.estimate	20,000	The upper limit on insert size that the pairing program will consider to calculate the size distribution.
min.insert.estimate	0	The lower limit on the insert size for the automatic insert range calculation.
pairing.mark.duplicates	true	Controls whether or not to track duplicates found (up to the limit specified by matching.max.hits). Allowed values are: • true • false
max.base.qv	40	The maximum value for a base quality value. Must be a value from 0 to 255.
matching.max.hits	100	The maximum number of hits for one tag for which the pipeline does rescue using the hits as anchors. Note: This value must match the value specified for the matching.max.hits parameter in your mapping.ini file.
reference	—	Full file name to the reference genome file.
pairing.color.qual.file.path.1	—	Optionally used instead of reads.results.dir.1 to specify the color quality files explicitly.
pairing.color.qual.file.path.2	—	Optionally used instead of reads.results.dir.2 to specify the color quality files explicitly.

Table 22 pairing.ini file parameter description *(continued)*

Parameter name	Default value	Description
pairing.correct.to	reference	Specifies which algorithm is used to correct color calls before converting to bases. Allowed values are: • missing: Replaces all inconsistent read-colors with '.'. These translate to 'x' in the base space representation, attribute 'b'. • reference: Replaces all read-colors annotated inconsistent (for instance, 'a' or 'b') with the corresponding reference color. • singles: Replaces all 'single' inconsistent colors (those annotated 'a' or 'b' and not adjacent to another 'b') with the corresponding reference color. Replaces all other inconsistent colors with '.'. • consistent: For each block of contiguous inconsistent colors, replaces all single inconsistent colors (those annotated 'a' or 'b' and not adjacent to another 'b') with the corresponding reference color. Replaces all other inconsistent colors with '.'. • qvThreshold: A scheme combining the other four algorithms, based on the specified qvThreshold.
pairing.tints	agy	Represents one or more single-tint annotations used to annotate color mismatches with respect to consistency with one, two, or three base variants. Allowed values are a string of one or more of: • a: Isolated single-color mismatches (grAy). • g: Color position that is consistent with an isolated one-base variant (for example, a SNP). • y: Color position that is consistent with an isolated two-base variant.
pairing.library.name	—	The User Library prefix used in the LB attribute of the BAM file. Note: Accepts any characters except tab and hyphen.
pairing.output.filter	primary	Determines alignments in the output BAM file. Allowed values are: • none: all, no filtering; • primary: only those marked as primary alignments.

Paired-end pairing.ini file example

This section describes an example pairing.ini file for paired-end analysis.

To run a paired-end pairing analysis, ensure you follow these steps in your pairing.ini file:

- Remove the pairing.run parameter or comment it out (with an initial '#' character):

  `#pairing.run=1`

  The setting `pairing.run=1` applies only to mate-pair pairing runs.
- Include the paired-end-pairing.run parameter, and set it to "1":

  `paired-end-pairing.run=1`

  Do not include both `pairing.run = 1` and `paired.end.pairing.run = 1` in the same configuration file. This causes pairing to run in two parallel jobs.
- Check the parameters appearing in Table 23 on page 164. Table 23 lists the parameters whose usage is different between mate-pair and paired-end pairing runs. (See "Mate-pair pairing.ini file example" on page 154 for an example mate-pair pairing.ini file.)

The following is an example of a paired-end pairing.ini file:

```
#           To include some common variables.
import ../../globals/global.ini
#Reference genome file name.
reference = ${reference.dir}/ch11_12_validated.fasta
reads.result.dir.1 = ${base.dir}/F3/reads
reads.result.dir.2 = ${base.dir}/F5/reads

##
*************************************************************
##   pairing
## ************************************************************

# mandatory parameters
# --------------------
# Parameter specifies whether to run or not pairing pipeline.
[1: to run, 0:to not run]
paired-end-pairing.run = 1

# Mapping output directories
mate.pairs.tagfile.dirs = ${base.dir}/F3/outputs/
s_mapping,${base.dir}/F5/outputs/s_mapping

pairing.output.dir = ${output.dir}/pairing

# optional parameters
# -------------------

# Selects a set of parameters for indel search: 1: Deletions to
11 [allowed values: 0, 1]
#indel.preset.parameters = 1

# Max Base QV. - The maximum value for a base quality value
```

```
#max.base.qv = 40

# Minimum Insert - Minimum insert size defining a good mate. If
this is not set the code will attempt to measure the best value
#insert.start =

# Maximum Insert - Maximum insert size defining a good mate. If
this is not set the code will attempt to measure the best value
#insert.end =

# Rescue Level - "Usually 2 * the mismatch level
#mate.pairs.rescue.level = 4

# Pairing statistics file name
#mates.stats.report.name = pairingStats.stats

# Max Hits for Indel Search
#indel.max.hits = 10

# Maximum Hits
#matching.max.hits = 100

# Mapping Mismatch Penalty
#mapping.mismatch.penalty = -2.0

# Parameter specifies the alignment size of the anchor region
#pairing.anchor.length=25

# Minimum Non-mapped Length for Indels
#indel.min.non-matched.length = 10

# Rescue Level for Indels - Default for 50mers,3 for 35mers, and
2 for 25mers.
#indel.max.mismatches = 5

# Use template Rescue File For Indels
#use.template.rescue.file = true

# Max mismatches in indel search for tag 1
#pairing.indel.max.mismatch.tag1 = 5

# Max mismatches in indel search for tag 2
#pairing.indel.max.mismatch.tag2 = 2

# Pair Uniqueness Threshold
#pair.uniqueness.threshold = 10.0

# Maximum estimated insert size
#max.insert.estimate = 20000

# Minimum estimated insert size
#min.insert.estimate = 0
```

```
# Primer set - Use this only when both directories specified by mate.pairs.tagfile.dirs are the same. Then the files must have these strings
# immediately before the .csfasta, if present, or the .ma extension.
# primer.set = F3,F5-P2

# Mark PCR and optical duplicates
#pairing.mark.duplicates = false

# Color quality file path 1 - Color quality file path for first tag. Use instead of reads directories.
#pairing.color.qual.file.path.1 =

# Color quality file path 2 - If either file path is explicitly set, both must be set
#pairing.color.qual.file.path.2 =

# Annotations: How to correct color calls -
#Specifies how to correct the color calls.
# 'missing' - Replaces all inconsistent read-colors with '.'. These will translate to 'x' in the base space representation, attribute 'b'.
# 'reference' - Replaces all read-colors annotated inconsistent (i.e., 'a' or 'b') with the corresponding reference color.
# 'singles' - Replaces all 'single' inconsistent colors (i.e., those annotated 'a' or 'b' and not adjacent to another 'b') with the corresponding
#   reference color. Replaces all other inconsistent colors with '.'.
# 'consistent' - For each block of contiguous inconsistent colors, replace all single insistent colors
#   (i.e., those annotated 'a' or 'b' and not adjacent to another 'b') with the corresponding reference color. Replace all other inconsistent
# colors with '.'.
# 'qvThreshold' - A scheme combining the four above choices, based on the specified qvThreshold. (--correctTo: default is missing)
#pairing.correct.to = reference

# Single-tint annotation - Represents any number of single-tint annotations.
# 'a' - Isolated single-color mismatches (grAy).
# 'g' - Color position that is consistent with an isolated one-base variant (e.g., SNP).
# 'y' - Color position that is consistent with an isolated two-base variant.
#    (default is agy if not specified.)
#pairing.tints = agy

# User Library prefix - Prefix for LB attribute of BAM file. Accepts any characters except tab and hyphen
#pairing.library.name =
# Parameter specifies the path to the ma result file of F3
mapping
#pairing.first.mapping.file=

# Parameter specifies the path to the ma result file of R3
mapping
#pairing.second.mapping.file=

# Parameter specifies filter for the records in the output  bam
file ['primary' specifies only primary alignments, 'none']
# Default value when not specified is 'primary'
#pairing.output.filter=primary

##################################
##
##  temp files and folders keep
##  # don't keep temp files and folders for clean run ...   make
it =1, if you want them to be deleted.
#job.cleanup.temp.files = 0
```

Paired-end pairing parameters

Table 23 lists pairing parameters which are either unique to paired-end runs or which have different default values or allowed values, compared to the corresponding parameter in mate-pair pairing (described in "Mate-pair pairing.ini file parameter descriptions" on page 157.

Table 23 Pairing parameters for paired-end pairing runs

Parameter name	Default value	Description
paired-end-pairing.run	—	Determines whether or not to run the paired-end pairing analysis. Allowed values are: • 0: Do not run the analysis. • 1: Run the analysis.
indel.preset.parameters	1	Selects a set of parameters for indel search: • 0: Do not perform an indel search. • 1: Searches for deletions to 11, insertions to 3.
primer.set		Allowed values are: • F3,F5 • F3,F5-P2 • F3,F5-BC Note: For paired-end-pairing, the value F3,R3 is not allowed.

Parameter name	Default value	Description
pairing.mark.duplicates	false	Controls whether or not to track duplicates found (up to the limit specified by matching.max.hits). Allowed values are: • true • false
pairing.indel.max.mismatch.tag2	2	Maximum number of mismatches allowed for indel/gap alignments on the F5 tag.

Run resequencing pairing

This section explains how to run pairing from the command line. The resequencing pairing run is performed automatically if you click Map Data in the web browser.

Complete the prerequisites

1. Complete the applicable prerequisites described in Chapter 3, "Before you Begin" on page 35.
2. Login to the BioScope™ Software cluster. Update the pairing.ini file with information that applies to the pairing pipeline that you plan to run.

Run Pairing from the command line

Although several different software programs are involved in the run, a single command generates all of the related programs required to complete the run. The *.plan file that is specified in the command syntax controls the order in which BioScope™ Software runs the related programs.

Start the run

1. Connect to the BioScope™ Software cluster and login with a user ID that has write privileges on all of the directories that BioScope™ Software uses when the tool runs.
2. At a command prompt, enter:

   ```
   bioscope.sh -l filename.log filename.plan
   ```

Do not log out of the BioScope™ Software cluster.

Check the run status from the command line

1. Navigate to the log directory that is defined in the pairing.ini file. For example, you might enter:

   ```
   cd /data/results/tertiary/cnv/log
   ```

2. Open `bioscope.`*`yyyymmddhhmmss`*`.log`.
3. Scroll to the end of the file.

The run is complete if you see an entry similar to:

```
15 Apr 2010 03:16:32,537  INFO [main] PluginJobManager:130 -
>>>> END of PluginJobManager >>>> date DURATION=4 minutes 33
secs
```

```
15 Apr 2010 03:16:32,537  INFO [main] EventTransportFactory:129
- Closing JMS connection and session
```

Pairing results file formats

For information about the *.bam file that is generated by pairing, see Appendix A, “File Format Descriptions” on page 295.

FAQs – Pairing

1

How is the uniqueness of a pair determined?

A pair of reads is unique when there is exactly one good AAA pair. With the advent of local alignment, the likelihood of finding only one good pair is decreased. As a result, a different heuristic is used to determine "uniqueness".

Consider an alignment of length L with M mismatches. If the local score is defined as $L+(m-1)M$, where $m<0$ is the mismatch penalty, then the score of each good pair candidate is the sum of its two constituent local scores.

A pair of tags is considered unique if it has only one good pair, or if the score of the best pair is at least X greater than the score of the second best pair. X is specified by the `pair.uniqueness.threshold` key, and has a default value of 10.0. The key is defined in the pairing.ini file. See Table 22 on page 157 for a description of all pairing.ini file parameters.

2

What do the 3-letter pair classifications mean?

Table 24 lists a summary of genomic code classifications for mate-pair pairing runs. See Table 25 on page 168 for a summary of genomic code classifications for paired-end runs.

Table 24 Genomic code classifications, for mate-pair pairing

Class name	Strand	Orientation	Distance	Preference
AAA	Same	R3 to F3	Normal	1
AAB	Same	R3 to F3	Too small	2
AAC	Same	R3 to F3	Too big	2
ABA	Same	F3 to R3	Normal	2
ABB	Same	F3 to R3	Too small	3
ABC	Same	F3 to R3	Too big	3
BAA	Different	Outward	Normal	3
BAB	Different	Outward	Too small	4
BAC	Different	Outward	Too big	4
BBA	Different	Inward	Normal	3
BBB	Different	Inward	Too small	4
BBC	Different	Inward	Too big	4

The first letter determines whether the two tags match the same strand:

- A=same
- B=different

The third letter determines if the distance between two hit positions is within the correct range:

- A=correct
- B=too small
- C=too big

The middle letter has different meanings, depending on whether the first letter is A or B. If the first letter is A, the second letter indicates if the order of the two tags is correct:

- A=correct order and means R3 is up stream of F3
- B=wrong order and means F3 is upstream of R3.

When the two tags are on different strands, the second letter indicates whether the two tags are pointing toward or away from each other

- A=toward each other
- B=away from each other

In addition, C** classification is assigned to a pair of reads that have one hit each, but are on different chromosomes. If both F3 and R3 tags are present, but only one has a unique hit while the other one is unmapped, the pair is labeled D**. Finally, if one tag from a pair has a unique hit but the other one is missing, as opposed to unmapped, then the pair is classified as E**.

The Preference value indicates likelihood of the variation can occur in nature. Larger values indicate the variation is less likely to occur in nature.

Table 25 shows the classifications for paired-end runs.

Table 25 Genomic code classifications, for paired-end pairing

Class name	Strand	Orientation	Distance	Preference
AAA	Different	Inward	Normal	1
AAB	Different	Inward	Too small	2
AAC	Different	Inward	Too big	2
ABA	Different	Outward	Normal	2
ABB	Different	Outward	Too small	3
ABC	Different	Outward	Too big	3
BAA	Same	F3 to F5	Normal	3
BAB	Same	F3 to F5	Too small	4
BAC	Same	F3 to F5	Too big	4
BBA	Same	F5 to F3	Normal	3
BBB	Same	F5 to F3	Too small	4
BBC	Same	F5 to F3	Too big	4

3

What information is contained in the pairing.stats file?

The pairing.stats file contains the following five sections:

Section 1

Section 1 of the pairing.stats file provides a distribution of three-letter pairing classifications among uniquely mapped reads (see Table 26).

Table 26 Distribution of three-letter pairing classifications

Category	Count	Percent	Cumulative	Cumulative Percentage
AAA	188425	53.67%	188425	53.67%
AAB	299	0.09%	188724	53.75%
AAC	6334	1.80%	195058	55.56%

Table 26 Distribution of three-letter pairing classifications

Category	Count	Percent	Cumulative	Cumulative Percentage
ABA	3	0.00%	195061	55.56%
ABB	5	0.00%	195066	55.56%
ABC	6262	1.78%	201328	57.34%
ABA	3	0.00%	195061	55.56%

Section 2

Section 2 of the pairing.stats file provides a detailed analysis of mate-pairs in the sample and in the proportion where each end is mapped, unmapped or missing in Pairing Statistics (see Table 27).

Table 27 Detailed analysis of mate-pairs

F3	R3	Count	Percentage
Mapped	Mapped	242072	42.98%
Mapped	Unmapped	45984	8.17%
Unmapped	Mapped	27009	4.80%
Unmapped	Unmapped	127320	22.61%

Section 3

Section 3 of the pairing.stats file provides a detailed summary of mapping for F3 tags. Alignments are grouped by the length of the hit, as well as the number of mismatches in the mismatch report for F3 (see Table 28)

Table 28 Detailed summary of mapping for F3 tags

Alignment Length	Mismatches	Count	Percent	Cumulative	Cumulative Percentage
49	0	13,933	6.51%	13,933	6.51%
49	1	11,011	5.15%	24,944	11.66%
49	2	8,787	4.11%	33,731	15.77%
49	3	6,402	2.99%	40,133	18.76%
49	4	4,365	2.04%	44,498	20.80%
49	5	2,865	1.34%	47,363	22.14%
49	6	1,553	0.73%	48,916	22.86%
49	7	751	0.35%	49,667	23.22%
49	8	315	0.15%	49,982	23.36%
49	9	121	0.06%	50,103	23.42%

Table 28 Detailed summary of mapping for F3 tags

Alignment Length	Mismatches	Count	Percent	Cumulative	Cumulative Percentage
49	10	38	0.02%	50,141	23.44%
49	11	7	0.00%	50,148	23.44%
48	0	1,799	0.84%	51,947	24.28%
48	1	2,270	1.06%	54,217	25.34%
48	2	2,205	1.03%	56,422	26.37%
48	3	1,845	0.86%	58,267	27.23%
48	1	2,270	1.06%	54,217	25.34%

Section 4

Section 4 of the pairing.stats file provides a detailed summary of mapping for R3 tags: Alignments are grouped by the length of the hit, as well as the number of mismatches in the mismatch report for R3 (see Table 29).

Table 29 Detailed summary of mapping for R3 tags

Alignment Length	Mismatches	Count	Percent	Cumulative	Cumulative Percentage
49	0	1,760	0.82%	1,760	0.82%
49	1	2,542	1.19%	4,302	2.01%
49	2	2,624	1.23%	6,926	3.24%
49	3	2,354	1.10%	9,280	4.34%
49	4	1,737	0.81%	11,017	5.15%
49	5	1,298	0.61%	12,315	5.76%
49	6	866	0.40%	13,181	6.16%
49	7	462	0.22%	13,643	6.38%
49	8	252	0.12%	13,895	6.49%
49	9	116	0.05%	14,011	6.55%
49	10	34	0.02%	14,045	6.56%
49	11	9	0.00%	14,054	6.57%
49	12	1	0.00%	14,055	6.57%
48	0	638	0.30%	14,693	6.87%
48	1	1,181	0.55%	15,874	7.42%
48	2	1,396	0.65%	17,270	8.07%

Section 5

Section 5 of the pairing.stats file provides a mapping summary of the *combined* alignment length and number of mismatches for each F3/R3 mate pair: A mapping summary with results for F3/R3 mate pairs is in the Mismatch report for combined (see Table 30).

Table 30 Mapping summary of the combined alignment length and number of mismatches (F3/R3 mate-pair)

Alignment Length	Mismatches	Count	Percent	Cumulative	Cumulative Percentage
98	0	245	0.11%	245	0.11%
98	1	458	0.21%	703	0.33%
98	2	517	0.24%	1,220	0.57%
98	3	571	0.27%	1,791	0.84%
98	4	470	0.22%	2,261	1.06%
98	5	418	0.20%	2,679	1.25%
98	6	330	0.15%	3,009	1.41%
98	7	264	0.12%	3,273	1.53%
98	8	196	0.09%	3,469	1.62%
98	9	140	0.07%	3,609	1.69%
98	10	98	0.05%	3,707	1.73%
98	11	52	0.02%	3,759	1.76%
98	12	35	0.02%	3,794	1.77%
98	13	22	0.01%	3,816	1.78%

11 Run the Find SNPs Tool

This chapter covers:

Find SNPs algorithm description

BioScope™ Software uses the diBayes tool to find Single Nucleotide Polymorphisms (SNPs). The diBayes package performs independent SNP analysis at each position in the reference, using either a Bayesian or Frequentist algorithm (see Figure 48 on page 175 and Figure 49 on page 175).

Frequentist algorithm

The Frequentist algorithm is used when coverage is high for a given position, for example, 60x. Given the assumption that the errors follow a Poisson distribution, the Frequentist algorithm calculates the probability of a null hypothesis that the observed valid dicolor mismatches are errors. If the probability of the null hypothesis is too low, the hypothesis is rejected and the position is called a SNP.

Bayesian algorithm

The Bayesian algorithm is used when the coverage of the position is not too high. When the Bayesian algorithm is applied, the posterior probability of the existence of a heterozygote or a non-reference homozygote at the position in question is evaluated. Based on the probability of SNP evidence being a mis-color call, position error, or probe error, the Bayesian algorithm uses the prior probability of the position being a heterozygote and the probability of observation being correct. The probabilities are calculated from the quality values of color-calls, the frequencies of dicolor read mismatches as a function of positions in a read, or from the frequencies of mismatches as the function of 6-mer probe prefix, respectively.

For more information about the Bayesian algorithm, visit the SOLiD™ Software Community at

solidsoftwaretools.com.

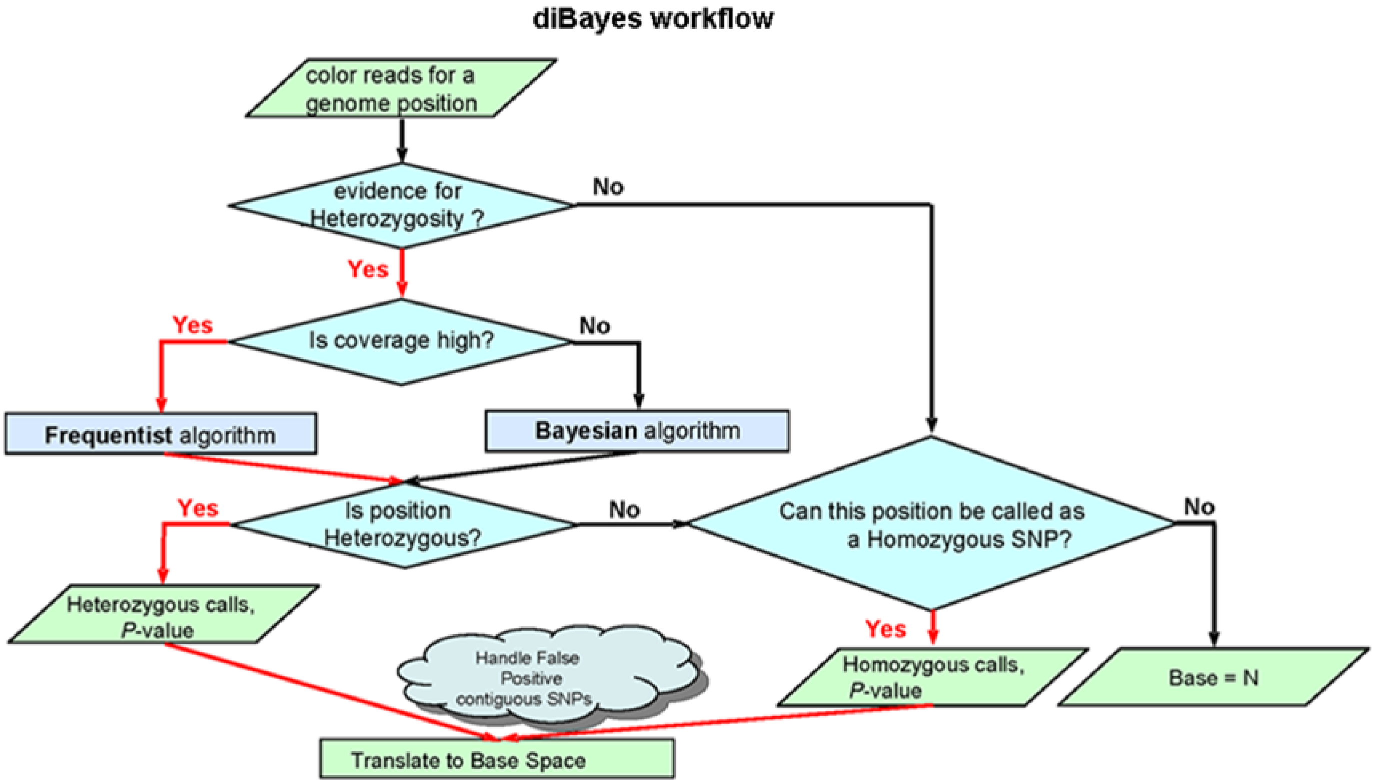

diBayes workflow
color reads for a genome position
evidence for Heterozygosity ?
No
Yes
Is coverage high?
Yes
No
Frequentist algorithm
Bayesian algorithm
Is position Heterozygous?
Yes
No
Can this position be called as a Homozygous SNP?
No
Yes
Heterozygous calls, P-value
Handle False Positive contiguous SNPs
Homozygous calls, P-value
Base = N
Translate to Base Space

Figure 48 diBayes algorithm flowchart

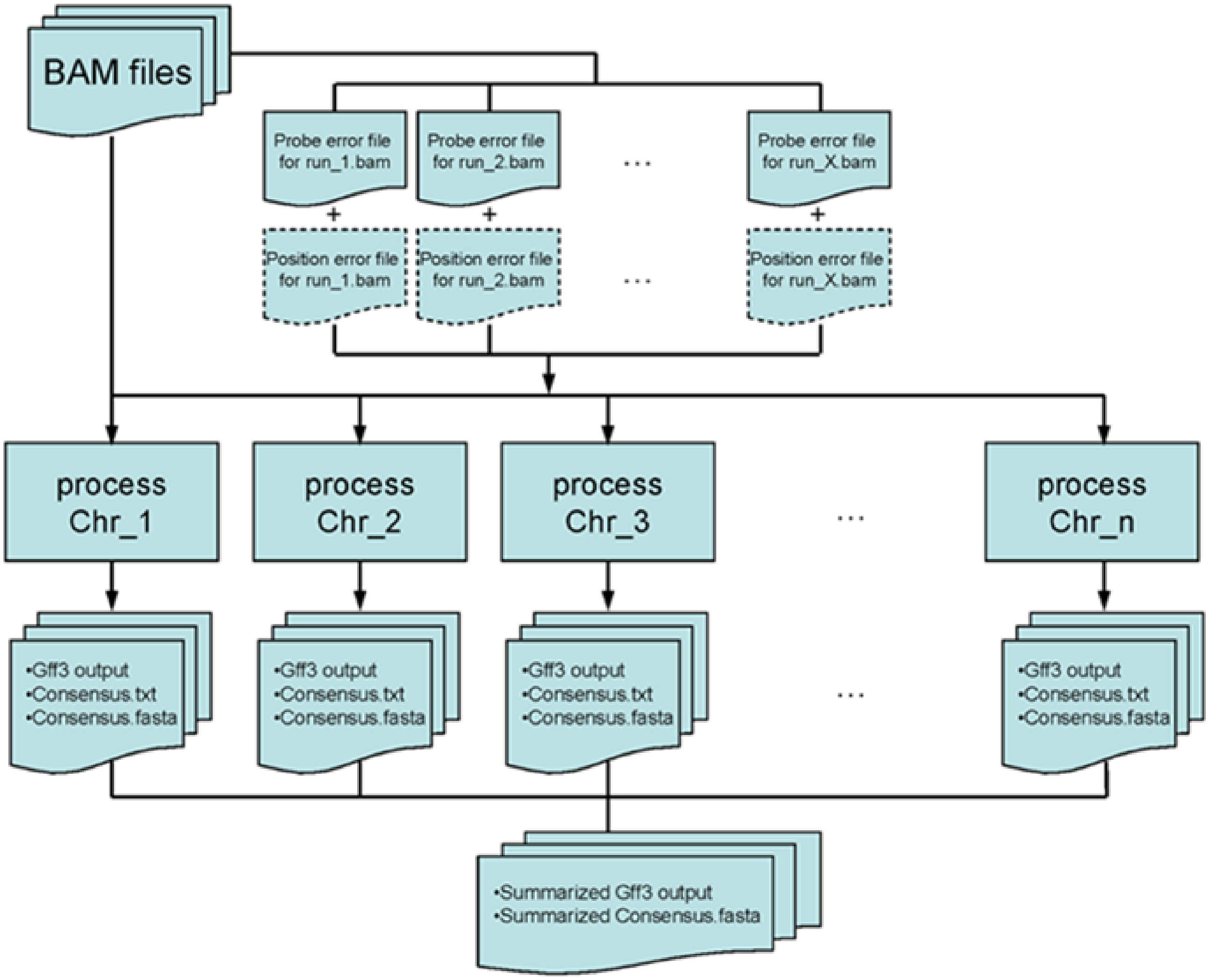

BAM files
Probe error file for run_1.bam
Probe error file for run_2.bam
. . .
Probe error file for run_X.bam
+
+
+
Position error file for run_1.bam
Position error file for run_2.bam
. . .
Position error file for run_X.bam
process Chr_1
process Chr_2
process Chr_3
. . .
process Chr_n
•Gff3 output
•Consensus.txt
•Consensus.fasta
. . .
•Summarized Gff3 output
•Summarized Consensus.fasta

Figure 49 diBayes data flow

Figure 49 shows how diBayes can take multiple *.bam files that are sorted by genome positions as input to make SNP calls at positions with coverage. Running diBayes requires the position error and probe error files of each *.bam file. The position error files are generated by the position_error tool. The probe error files are calculated within diBayes. In a multi-CPU computing cluster, diBayes automatically distributes parallel jobs for individual chromosomes to different computing nodes. The final results for the whole genome are merged after the individual jobs are complete.

Configure SNP parameter settings

The diBayes input files include the reference file, the *.bam file, and their corresponding position error and probe error files (see Table 31 on page 176).

Table 31 diBayes input file description

Input File	Description
Reference genome	The sequence to which the reads are aligned and mapped. The recommended format for the reference file is the *.fasta format, for example, chr20.validated.fasta. Note: When you run the SNPs tool, you must use the same reference that was used for mapping. The reference sequence might have multiple chromosomes or contigs, and might contain IUB codes at positions of known SNPs. A heterozygote is called with fewer reads providing evidence at these positions if the reference sequence contains IUB codes, because the prior probability of a heterozygote existing at this position is higher.
F3 (R3/F5) position error file	F3 (R3/F5) position error files are tab-delimited text files created during secondary analysis. The files record the frequencies of dicolor mismatches between reads and the reference at different positions in a read. For fragment runs, diBayes only requires an F3 position error file. Both F3 and R3(F5) position error files are created for mate-pair (paired-end) runs, and both are required for running diBayes.
F3 (R3/F5) probe error file	F3 (R3/F5) probe error files are tab-delimited text files that record the frequencies of dicolor mismatches between the reads and the reference as a function of different 6-mer probes. BioScope™ Software calculates the probe error files. Reads only have the F3 tag for fragment runs. Thus, diBayes requires only an F3 probe error file. Both F3 and R3/F5 probe error files are created for mate-pair (paired-end) runs, and both are required to run diBayes. Different SOLiD™ system runs of the same sample might generate different F3 (R3/F5) probe error files for each run.
*.bam file	The *.bam input file is generated by BioScope™ Software at the end of mapping or pairing.

Input file parameters

Set the following keys in the snp.ini file to configure the input file parameters:

```
- reference= // location of the reference file.
- input.file.info= // BAM input files and position error files.
```

The input.file.info parameter is a comma-separated list of the input sets in the following format:

```
- input.file.info= \
file1.bam:run_type:f3-positionerror[:r3/f5 positionerror],\
file2.bam:run_type:f3-positionerror[:r3/f5 positionerror],\
file3.bam:run_type:f3-positionerror[:r3/f5 positionerror],...
```

where `run_type` can be:

- `0 (fragment)`
- `1 (mate-pair)`
- `2 (paired-end run)`

For example, you might have three *.bam files; `run1_frag.bam`, `run2_MP.bam`, and `run3_PE.bamruns`, which are mapping and pairing results of a fragment run, a mate-pair run and a paired-end run, respectively. Enter the following input.file.info parameter as shown below:

```
input.file.info= \
/run1/path/run1_frag.bam:0:f3_positionerror_run1_frag.txt,\
/run2/path/run2_MP.bam:1:f3_positionerror_run2_MP.txt:\
r3_positionerror_run2_MP.txt,\
/run3/path/run3_PE.bamruns:2:f3_positionerror_run3_PE.txt:\
f5_positionerror_run3_PE.txt
```

Output directory parameters

Enter the parameters that define the output directory structure (see Table 32). You must define the output directory parameters before you can use the SNP tool.

Table 32 SNP output directory parameters

dibayes.output.dir	The name of the folder that will contain the final diBayes output files.
dibayes.working.dir	The name of the folder that will contain the temporary and intermediate files.
dibayes.log.dir	The name of folder that will contain the log files
dibayes.output.prefix	The prefix of the output file and the subfolder name within the previous three folders.

IMPORTANT! The name chosen for the dibayes.output.prefix variable is used as the prefix for the output file names generated when the diBayes tool runs, and as the output directory for the final results. To prevent overwriting files, change the `dibayes.output.prefix` in the diBayes.ini file before each run.

Note: Do not merge the *.bam files before running the SNPs tool. The tool can accept multiple *.bam files of different run types as its input. The tool rejects all merged *.bam files.

Mandatory algorithm settings

Define the mandatory algorithm parameters (see Table 33). You must define the algorithm parameters before you can use the SNPs tool.

Table 33 diBayes.ini file mandatory algorithm settings

Parameter	Description
maximal.read.length	Sets the largest read length of multiple runs.
call.stringency	Defines the SNP call stringency.

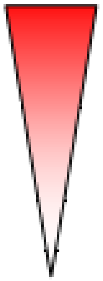
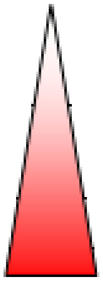
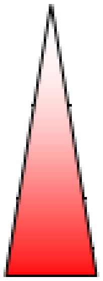
The call stringency parameter

The `call.stringency` parameter controls the SNP calling results. Each setting described in represents a different combination of filters (see Table 34). Select one of five values to set the call stringency parameter. Figure on page 179 shows the call stringency values described in Table 34.

Table 34 Call stringency parameter options

Parameter	Description
highest	Select the “highest” value with data sets that have > 80*x* coverage, or for scenarios in which very low false-positive tolerances are allowed.
high	Select the “high” value for data with 20 to 80*x* coverage. You can also use the “high” setting with lower coverage sets when very low false-positive tolerances are allowed.
medium	The “medium” setting is optimized for data sets with 1*x* to 25*x*. You can use the setting on datasets with higher coverage if you want to have more SNPs but with more false positives. The main difference between the high and medium settings is that the medium setting does not require coverage of the alleles on both strands. Use medium if you expect differences in the coverage of the two alleles on both strands of DNA, such as transcriptome data, or certain kinds of DNA enrichment data.
low	The “low” setting has the least stringency. When you select the “low” setting, an SNP can be called even though only a single observation of the non-reference allele is seen. When you select the “low” setting, the SNP call has a higher false-positive rate.

Table 35 Four empirical call.stringency settings for diBayes

call.strigency	stringency	SNPs	false positives	Recommended coverage	Comments
highest				>80x	Recommend when very low false positive tolerance is allowed.
high				20x ~ 80x	Requires the allele on both strands.
medium				1x ~ 25x	No both-strand requirement.
low				—	Very aggressive.

Enable the het.skip.high.coverage filter

Enable the `het.skip.high.coverage` filter to skip false-positive SNP positions with high coverage. If you enable the het.skip.high.coverage filter, the SNP position is skipped if the coverage of a position is too high compared to the median of the coverage distribution of all positions. The filter is disabled by default. Enable the `het.skip.high.coverage` filter for whole genome resequencing. Disable the filter for transcriptome or target resequencing.

Set the reads.min.mapping.qv parameter

Modify the default setting of zero in the `reads.min.mapping.qv` filter if mapping and pairing quality is a concern. The advanced settings for diBayes are optional. These filters provide additional freedom for different datasets and applications.

IMPORTANT! If you want to use the diBayes method to analyze multiple slides which mix fragment, mate-pair, and paired-end data, create a separate *.bam file for each slide, and input all of the *.bam files in the diBayes.ini file (see "Input file parameters" on page 176). Do not concatenate *.bam files from different run types.

dibayes.ini file example

The following section shows a typical example of the dibayes.ini file. For a description of the dibayes.ini file parameters, see Table 36 on page 184.

IMPORTANT! Before you begin a run, you must verify the settings for each parameter that is highlighted in **bold** in the *.ini file example shown in the next section.

```
## This is a configuration file for diBayes.
import ../globals/global.ini

## *********************************************
## mapping parameters
## *********************************************
```

```
mapping.output.dir=${output.dir}/mapping${primer.set}/s_mapping


## ********************************************
## positionErrors parameters
## ********************************************
position.errors.output.dir=${output.dir}/positionErrors/

## ********************************************
## diBayes parameters
## ********************************************


# mandatory parameters
# --------------------

# Parameter to specify whether to run Mutation Pipeline.
[Options: 1 - Run, 0 - Don't run].
dibayes.run=1

# Parameter specifies the full path to location of directory
where to write diBayes output files.
dibayes.output.dir=${output.dir}/diBayes/DB_OUT

# Parameter specifies the full path to location of directory in
which to place temporary working files
dibayes.working.dir = ${temp.dir}/dibayes

# Parameter specifies the full path to the log directory
dibayes.log.dir = ${log.dir}/dibayes

# Parameter specifies the name of subdirectory in the output
folder and the Name of the experimentprefix of the output files
dibayes.output.prefix = test_SNP

# Parameter specifies the reference sequence fasta file with
full path
reference=${reference.dir}/
DH10B_WithDup_FinalEdit_validated.fasta

# Parameters specifies colon-separated list of the input sets in
the format:
# file-full-path:mate-pair-flag:f3-position-err-file:[r3-
position-err-file]
input.file.info=${base.dir}/outputs/pairing/F3-R3-
Paired.bam:1:${base.dir}/outputs/position-errors/F3-R3-
Paired_F3_positionErrors.txt:${base.dir}/outputs/position-
errors/F3-R3-Paired_R3_positionErrors.txt


# Maximal read length (e.g. 50). Note: this program allows
# combining reads from sources with different read lengths.
maximal.read.length = 50
```

```
# The parameter the criteria to report SNPs. [Options:
highest|high|medium|low]
# Default value is 'medium' when not mentioned.
call.stringency = medium

# optional parameters
# Changes on the algorithm fine tuning parameters will override
the values that are preset by call.stringency setting.
# -------------------

# Polymorphism rate: Expected frequency of heterozygotes in the
population: for example, 0.001 in humans
#poly.rate = 0.001

# Parameter specifies to detect 2 Adjacent SNP's. [Options: 0 -
do not detect, 1 - detect].
#detect.2.adjacent.snps=0

# Parameter specifies whether to write fasta file or not.
[Options: 0 - Don't write fasta file, 1 - Write fasta file].
# Default value when not specified is 1.
write.fasta = 1

# Parameter specifies whether to write consensus_calls.txt.
[Options: 0 - Don't write, 1 - Write].
# Default value when not specified is 1.
write.consensus = 1

# Parameter specifies whether to compress consensus_calls.txt by
zipping it.  [Options: 0 - Don't ZIP, 1 - ZIP].
compress.consensus = 0

# Parameter specifies whether to clean up the temporary files.
[Options: 0 - Don.t clean, 1- Clean].
# Default value when not specified is 1.
#cleanup.tmp.files =1

# Parameter specifies not to call SNPs when the coverage of
position is too high comparing to the median of the coverage
distribution of all positions.
# NOTE: enable this filter for whole genome re-sequencing
application; disable (default) it for transcriptome or target
re-sequencing.
#het.skip.high.coverage=1

# Parameter specifies the minimum mapping/pairing quality value.
# Default value when not specified is 0
#reads.min.mapping.qv=8

# Parameter specifies the required minimum for color quality
value of non-reference allele to call a heterozygous SNP.
#het.min.nonref.color.qv=7
```

```
# Parameter specifies the required minimum for color quality
value of non-reference allele to call a homozygous SNP.
#hom.min.nonref.color.qv=7

# Parameter species the requirement that the novel allele be on
both strands :
# Parameter specifies the requirement that the novel allele is
present on both strands and
# statistically similarly represented on both strand for both
heterozygous and homozygous positionsSNPs.
# [Options: 0 - don't require, 1 - require]
#snp.both.strands = 0

# Parameter specifies the minimum required coverage to call a
heterozygous SNP.
# [Allowed Values: Integer, 1-n]
#het.min.coverage = 3

# Parameter specifies Mthe minimum number of unique start
position required to call a heterozgyote.
# [Allowed values: Integer, 1-n]
#het.min.start.pos = 3

# Parameter specifies the proportion of the reads containing
either of the two candidate alleles.
# Filters positions with high raw error rate.
# [Allowed values: Float, 0-1]
#het.min.ratio.validreads=0.65

# The less common allele must be at least this proportion of the
reads of the two heterozygote alleles.heterozygote.
# [Allowed values: Float, 0-1]
#het.min.allele.ratio=0.15

# Parameter specifies the Require at minimumleast 2 number of
reads of an apparently valid tricolor calls to pass through
filter to call 2 adjacent basesSNPs. i
# [Allowed Values: Integer]
#het.min.counts.tricolor=2

# Parameter specifies the required minimum coverage to call a
homozygous SNP.
#hom.min.coverage=3

# Parameter specifies the Mminimum number of unique start
position required to call a homozgyote.
# [Allowed values: Integer, 1-n]
#hom.min.start.pos=3

# Parameter specifies the required minimum coverage of candidate
allele to consider this genome position for a Hhomozygous call.
#hom.min.allele.count=3
```

```
# Parameter specifies whether or not to filter the reads with
indels.[Options: 0 - don't filter, 1 - filter].
# Default value when not specified is 1
#reads.no.indel=1

# Parameter specifies whether or not the reads to beare uniquely
mapped. [Options: 0 - don't require, 1 - require].
# Default value when not specified is 0
#reads.only.unique=0

# Parameter specifies the threshold of mismatch/alignment length
ratio.
# The reads whose mismatch/alignment length ratio is HIGHER than
this specified threshold will be filtered.
# [Allowed values: Float, 0 - 1, 1 - don't filer]
#reads.max.mismatch.alignlength.ratio=1.0

# Parameter specifies rtTthe threshold of alignment-length /
read-length ratio.
# The reads whose alignment-length/read-length ratio is are LESS
than this specified threshold will be filtered.
# [Allowed values: Float, 0 - 1, 0 - don't filer]
#reads.min.mismatch.alignlength.ratio=0.0

# Parameter specifies whether to include the reads that only
have one tag mapped (their mate tags are either unmapped or
missing.)
# [Options: 0 - don't include, 1 - include]
#reads.include.no.mate=0
```

Notes about Table 36:

- [1]Display in UI: Global-Global settings, Basic-Application Settings, Advanced - Advanced Setting
- [2]D=Default value
- [3]Required when runtype = 1 or 2 (mate-pair or paired-end)
- [4]The default values of keys in the Optional Algorithm tuning parameters section marked with [4] depend on the stringency settings. New user inputs override the original settings.
- [6]Keys starting with “snp” and marked with [6] are the filters for both homozygous and heterozygous SNP calls.
- [7]Keys starting “het.” and marked with [7] are the filters for both heterozygous SNP calls (positions).
- [8]Keys starting with “hom.” and marked with [8] are the filters for both homozygous SNP calls.
- [9]Keys starting with “reads.” and marked with [9] are the filters for the reads.

Table 36 SNP (diBayes) parameter description

Parameter name	**U[1]**	**Range and Default**	**Comment**
dibayes.run	No	0 - Don't run; 1 - Run (D[2])	Whether to run the mutation pipeline.
dibayes.output.dir	Global	String	The path to the directory to write the diBayes output files.
dibayes.working.dir	Global	String	The path to the directory to place the temporary files.
dibayes.log.dir	Global	String	The path to the directory to place the log files.
dibayes.output.prefix	Basic	String with no spaces	The prefix of the output files.
reference	Basic	String	Reference sequence *.fasta file with full path.

input.file.info	Basic	String	Comma-separated list of the input sets in the format:
			input.file.info= <*.bam file1>:<Run type (0, 1 or 2), 0-fragment, 1-mate-pair, 2-paired-end >:f3-position-err-file:[r3/f5-position-err-file][3],<*.bam file2>:<Run type (0, 1 or 2), 0-fragment, 1-mate-pair, 2-paired-end >:f3-position-err-file:[r3/f5-position-err-file][3]
			example:
			input.file.info= AA_frag.bam:0:f3-position-err-file, AA_matepair.bam:1:f3-position-err-file: r3-position-err-file

maximal.read.length	Basic	Integer	Maximal read length, for example, 50. Note: This program allows combining reads from sources with different read lengths.

call.stringency	Basic	highest;
		high (D);
		medium;
		low
			Defines the SNPs call stringency.

het.skip.high.coverage	Basic	0-Don't skip (D) 1- Skip	Do not call SNPs when the coverage of position is too high compared to the median of the coverage distribution of all positions. Note: Enable this filter for whole genome resequencing application. Disable it for whole transcriptome or target resequencing.
Optional Parameters			
poly.rate	Advanced	Float	Polymorphism rate: The expected frequency of heterozygotes in the population, for example, 0.001 in humans.

detect.2.adjacent.snps	Advanced	0 - Don't detect (D);
		1- detect;
			Detect two adjacent SNPs.

Table 36 SNP (diBayes) parameter description *(continued)*

Parameter name	U[1]	Range and Default	Comment

write.fasta	Advanced	0 - Don't write (D);
		1- write (D);
			Whether to write *.fasta files.

write.consensus	Advanced	0 - Don't write (D);
		1- write (D);
			Whether to write consensus_calls.txt files.

compress.consensus	Advanced	0 - Don't compress (D); 1 - Compress	Whether to zip the generated consensus files.
cleanup.tmp.files	No	0 - Don't clean; 1 - Clean (D)	Whether to clean up the temporary files.
Optional Algorithm tuning parameters[4]			
reads.min.mapping.qv	Advanced	Integer (0,default-100)	Requires that the mapping quality value of the read be higher than this minimum mapping/pairing qv.
het.min.nonref.color.qv	Advanced	Integer	Requires the non-reference allele to have at least this color quality value to call a heterozygous SNP.
hom.min.nonref.color.qv	Advanced	Integer	Requires the non-reference allele to have at least this color quality value to call a homozygous SNP.

snp[6].both.strands	Advanced	0=don't require;
		1 = require;
			Require that the novel allele is present on both strands and statistically similar represented on both strand for both heterozygous and homozygous SNPs,

het[7].min.coverage	Advanced	Integer	Require at least this coverage to call a heterozygous SNP.
het.min.start.pos	Advanced	Integer	The minimum number of unique start positions required to call a heterozygote.
het.min.ratio.validreads	Advanced	Float (0~1)	The proportion of the reads containing either of the two candidate alleles. Filters positions with high raw-error rate.
het.min.allele.ratio	Advanced	Float (0~1)	The less-common allele must be at least this proportion of the reads of the two heterozygous alleles.
het.min.counts.tricolor	Advanced	Integer	Requires at least two reads of an apparently valid tricolor to pass through the filter to call two adjacent SNPs.
hom[8].min.coverage	Advanced	Integer	Requires at least this coverage to call a homozygous SNP.
hom.min.start.pos	Advanced	Integer	The minimum number of unique start positions required to call a heterozygote.
hom.min.allele.count	Advanced	Integer	Requires at least this coverage of candidate allele to consider this genome position for a homozygous call.

Table 36 SNP (diBayes) parameter description *(continued)*

Parameter name	U[1]	Range and Default	Comment
reads.no.indel	Advanced	0-Don't filter; 1-Filter (D)	Filter the reads with indels.
reads.only.unique	Advanced	0-Don't require (D); 1-require	Requires the reads to be uniquely mapped. A very stringent filter.
reads[9].max.mismatch.alignlength.ratio	Advanced	Float (0~1, 0-Don't filter)	The threshold of mismatch/alignment length ratio. The reads whose mismatch/alignment length ratio is higher than this specified threshold will be filtered.
reads.min.alignlength.readlength.ratio	Advanced	Float (0~1, 0-Don't filter)	The threshold of alignment-length / read-length ratio. The reads whose alignment-length/read-length ratio is less than this specified threshold will be filtered.

reads.include.no.mate	Advanced	0-Don't include (D);
		1-include;
			Include the reads that only have one tag mapped (their mate tags are either unmapped or missing.)

Prepare to run the Find SNPs tool

Select the required input files

Before you can run the Find SNPs tool you must know the following information:

- The absolute path to the *.fasta file
- The absolute path to the *.bam file
- The absolute path to the F3 Position Error text file
- The absolute path to the R3/F5 Position Error text file

Complete the prerequisites

1. Complete the applicable prerequisites described in Chapter 3, "Before you Begin" on page 35.

2. Login to the BioScope™ Software cluster. Change to the working directory and update the dibayes.ini file with information that applies to the Find SNPs run. See "dibayes.ini file example" on page 179.

3. Create an output prefix. Output prefixes are case-sensitive. Use an underscore to separate terms in an experiment name. Example: Experiment_1.

4. Complete the resequencing mapping/pairing process on the primary data from the instrument.

Run the Find SNPs tool from the command line

Although several different software programs are involved in the experiment, a single command generates all of the related programs required to complete the experiment. The *.plan file that is specified in the command syntax controls the order in which BioScope™ Software runs the related programs.

Start the run

1. Connect to the BioScope™ Software cluster and login with a user ID that has write privileges on all of the directories that BioScope™ Software uses when the tool runs.
2. At a command prompt, enter:
   `bioscope.sh -l` *`filename`*`.log` *`filename`*`.plan`

Do not log out of the BioScope™ Software cluster.

Check the run status from the command line

1. Navigate to the log directory that is defined in the diBayes.ini file. For example, you might enter:
   ```
   cd /data/results/tertiary/diBayes/log
   ```
2. Open `bioscope.`*`yyyymmddhhmmss`*`.log`.
3. Scroll to the end of the file.

The run is complete if you see an entry similar to:

```
15 Apr 2010 03:16:32,537  INFO [main] PluginJobManager:130 -
>>>> END of PluginJobManager >>>> date DURATION=4 minutes 33
secs

15 Apr 2010 03:16:32,537  INFO [main] EventTransportFactory:129
- Closing JMS connection and session
```

Run the Find SNPs tool from the web interface

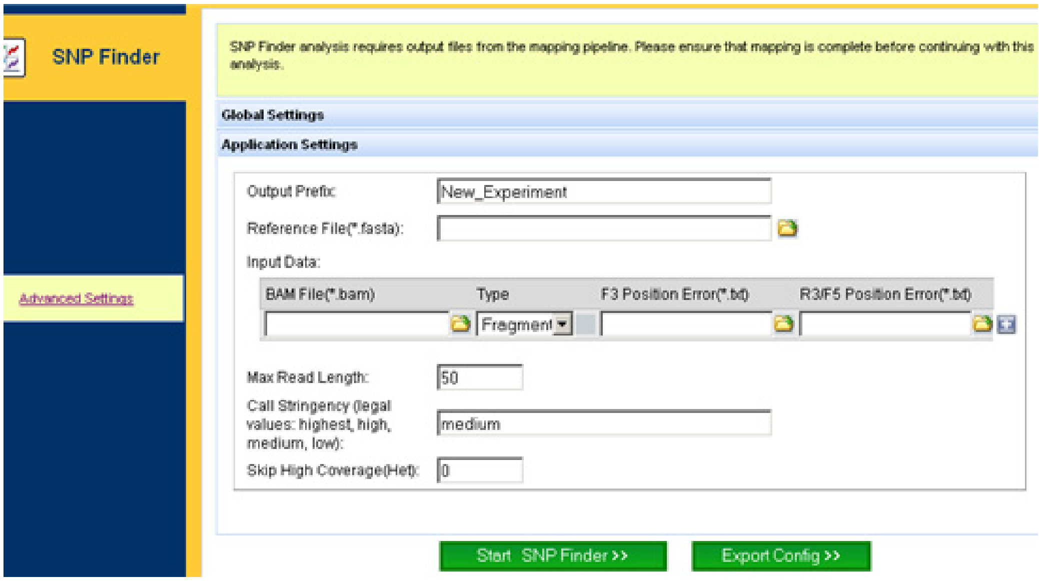
The instructions in this section assume the following system conditions:

- The Java Messenger, Tomcat, and Apache services are running on the BioScope™ Software cluster.
- You are using Internet Explorer versions 6 or 7 or Mozilla 3.0.1.
- You have planned the name of the diBayes output file prefix.
- Mapping and pairing is complete.

1. Launch a browser and enter the BioScope™ Software URL:
   http://<hostname>:8080/bioscope
2. Click **Find SNPs**.

The Find SNPs page has two windows and one link (see Figure 50).

- Global Settings
- Applications Settings
- Advanced Settings



SNP Finder

Advanced Settings

SNP Finder analysis requires output files from the mapping pipeline. Please ensure that mapping is complete before continuing with this analysis.

Global Settings

Application Settings

Output Prefix: New_Experiment

Reference File(*.fasta):

Input Data:

BAM File(*.bam)	Type	F3 Position Error(*.txt)	R3/F5 Position Error(*.txt)
	Fragment		

Max Read Length: 50

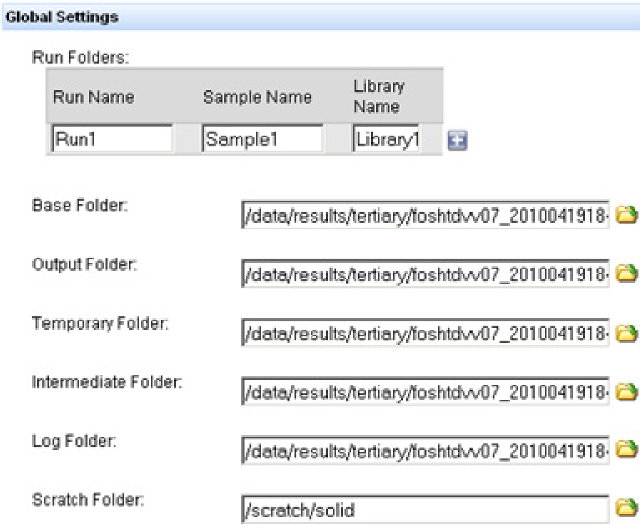
Call Stringency (legal values: highest, high, medium, low): medium

Skip High Coverage(Het): 0

Figure 50 Find SNPs web page example

Global Settings description

The Global Settings section displays the default values for the folders that BioScope™ Software creates for the files that result from the Find SNPs run (see Figure 51 on page 188). The section also has fields where you can enter the Run Name, Sample Name, and Library Name of the primary data that was exported to BioScope™ Software from the instrument.



Global Settings

Run Folders:

Run Name	Sample Name	Library Name
Run1	Sample1	Library1

Base Folder: /data/results/tertiary/foshtdvv07_2010041918

Output Folder: /data/results/tertiary/foshtdvv07_2010041918

Temporary Folder: /data/results/tertiary/foshtdvv07_2010041918

Intermediate Folder: /data/results/tertiary/foshtdvv07_2010041918

Log Folder: /data/results/tertiary/foshtdvv07_2010041918

Scratch Folder: /scratch/solid

Figure 51 Find SNPs Global Settings section example

Customize the default folder structure (optional)

The folders store the results files generated by each Find SNPs run. BioScope™ Software automatically creates the default folder structure for each Find SNPs run: `/data/results/tertiary/`*`headnode_yyyymmddhhmmss_x`*

Complete the following steps to change the default directory structure.

1. Click 📂 in the Base Folder field. The File Browser dialog appears.
2. In the **Look in** field, type the custom directory path, for example, `/home/data`
3. Click **Open**.
4. The folders reflect the updated directory structure.

Note: If you change the default directory structure, the Output, Temporary, Intermediate, and Log folders become subdirectories of the Base Folder.

Update the Run Folder settings (optional)

You can accept the default values in the Run Name, Sample Name and Library Name fields. In this context, “run” refers to the primary data that was exported to BioScope™ Software from the instrument.

To change the default values for the Run Folders:

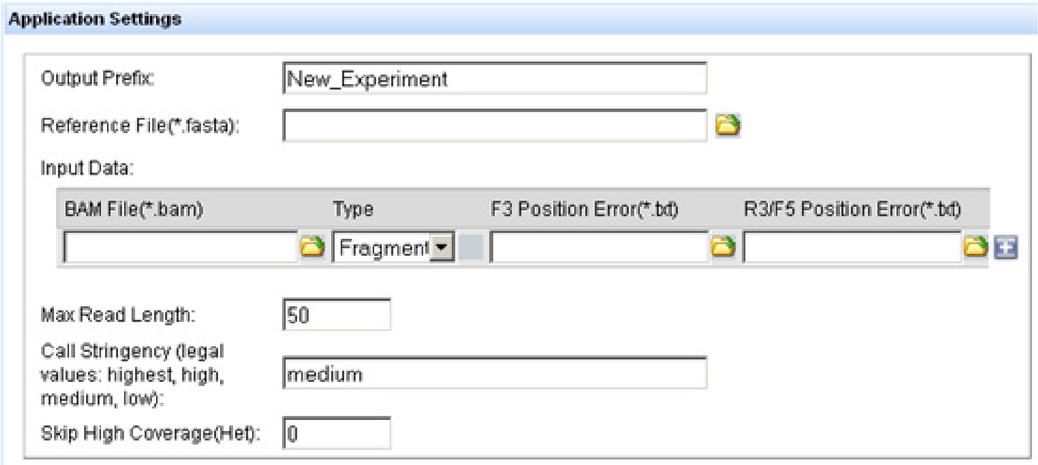
1. Enter the updated run name in the Run Name field.
2. Enter the updated sample name in the Sample Name field.
3. Enter the updated library name in the Library Name field.
4. Optional: Click ⊞ to add a row for a second run folder.
5. Optional: Enter a Run Name, a Sample Name and a Library Name in the new row.

Advanced Settings description

Click **Advanced Settings** to view the current default values defined by BioScope™ Software for the Find SNPs tool. Do *not* change any Advanced Settings unless instructed to by the BioScope™ Software administrator.

Application Settings description

In the Application Settings section (see Figure 52), you can accept or change the default setting for the Output Prefix and define the absolute paths to the *.fasta and input *.bam file(s). You also select the input data type and define the absolute paths to the F3 Position Error and R3/F5 Position Error files. The Export Config >> button is only used with the tool that processes barcoded libraries (see Appendix C, “Batch Analysis of Barcoded Library Data” on page 319).



Application Settings

Output Prefix: New_Experiment

Reference File(*.fasta):

Input Data:

BAM File(*.bam)	Type	F3 Position Error(*.txt)	R3/F5 Position Error(*.txt)
	Fragment		

Max Read Length: 50

Call Stringency (legal values: highest, high, medium, low): medium

Skip High Coverage(Het): 0

Figure 52 Find SNPs Application Settings window

Start the Find SNPs tool run

1. Change the default Output Prefix name or accept the default name.
2. Click 📂 in Reference File (*.fasta). The File Browser window appears.
3. Define the absolute path to the *.fasta file.
4. Click **Open**.
5. Click 📂 in the BAM File(*.bam) field. The File Browser window appears.
6. Define the directory path to the *.bam file.
7. Click **Open**.
8. Select the data type of primary run.
9. Click 📂 in F3 Position Error(*txt). The File Browser window appears.
10. Define the directory path to the F3 Position Error file.
11. Click **Open**.
12. Click 📂 in R3/F5 Position Error(*txt). The File Browser window appears.
13. Define the directory path to the R3/F5 Position Error file.
14. Click **Open**.
15. Optional: Click ⊞ to define a path to a second folder that contains a *.bam file and repeat steps 5 to 13.
16. Enter the Max Read Length.
17. Enter the Call Stringency setting.
18. Enter Skip High Coverage (Het), as follows:

- 0 for transcriptome and target resequencing data
- 1 for whole genome resequencing data

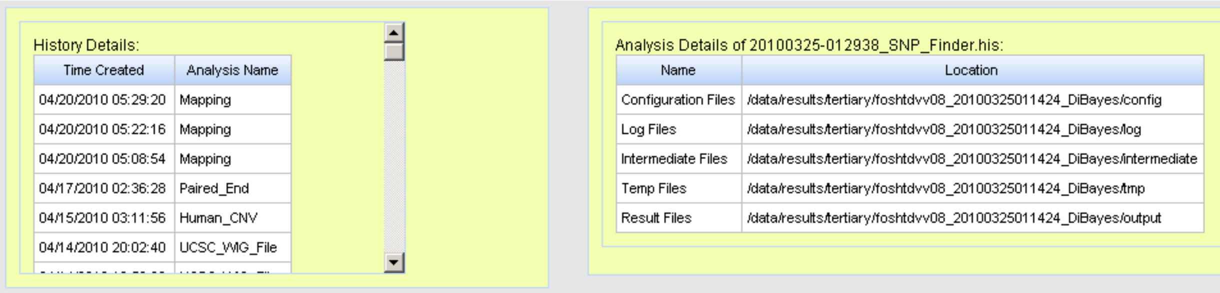
19. Click Start SNP Finder >> to start the analysis.

20. At the job submission dialog, click **OK** after you have verified the folder locations.

Check the status of the run from the web interface

1. Click History . The History window appears and the History Details table is displayed in the left pane. The History Details table shows the Time Created and Analysis Name for all runs performed on the BioScope™ Software cluster.

2. Scroll the History Details table and select a SNP_Finder run, based on the data in the Time Created column (see Figure 53).



History Details:

Time Created	Analysis Name
04/20/2010 05:29:20	Mapping
04/20/2010 05:22:16	Mapping
04/20/2010 05:08:54	Mapping
04/17/2010 02:36:28	Paired_End
04/15/2010 03:11:56	Human_CNV
04/14/2010 20:02:40	UCSC_WIG_File

Analysis Details of 20100325-012938_SNP_Finder.his:

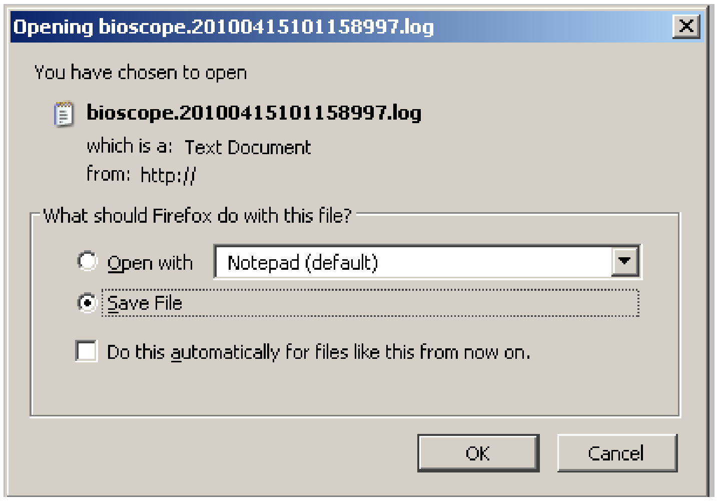
Name	Location
Configuration Files	/data/results/tertiary/foshtdvv08_20100325011424_DiBayes/config
Log Files	/data/results/tertiary/foshtdvv08_20100325011424_DiBayes/log
Intermediate Files	/data/results/tertiary/foshtdvv08_20100325011424_DiBayes/intermediate
Temp Files	/data/results/tertiary/foshtdvv08_20100325011424_DiBayes/tmp
Result Files	/data/results/tertiary/foshtdvv08_20100325011424_DiBayes/output

Figure 53 History details and analysis details for a Find SNPs tool run

3. Double-click the Log Files row in the Analysis Details table. The File Browser dialog opens. Click **Resend** if your browser displays a message.

4. Select the `bioscope.`*`yyyymmddhhmmss`*`.log` file.

5. Click **Download**.
   - Click **Open with** and click **OK** to view the log file in Notepad or select a different text editor.
   - Click **Save File** to copy the file to your workstation.



Opening bioscope.20100415101158997.log

You have chosen to open

bioscope.20100415101158997.log

which is a: Text Document
from: http://

What should Firefox do with this file?

○ Open with Notepad (default)

● Save File

□ Do this automatically for files like this from now on.

OK Cancel

Figure 54 Log file download page example

6. Scroll to the end of the file.

The run is complete if you see an entry similar to:

```
15 Apr 2010 03:16:32,537  INFO [main] PluginJobManager:130 -
>>>> END of PluginJobManager >>>> date DURATION=4 minutes 33
secs

15 Apr 2010 03:16:32,537  INFO [main] EventTransportFactory:129
- Closing JMS connection and session
```

Find SNPs output file formats

As shown in Figure 49 on page 175, in the diBayes output folder (defined by dibayes.output.dir), there are multiple output subfolders (e.g. chr_1, chr_2, ... chr_n), corresponding to individual chromosomes chromosome/contigs. Each subfolder usually contains 4 files:

- `<dibayes.output.prefix>_SNP.gff3`
- `<dibayes.output.prefix>_Consensus_Calls.txt`
- `<dibayes.output.prefix>_Consensus_Basespace2.fasta`
- `<dibayes.output.prefix>_quartiles.txt`

The file `<dibayes.output.prefix>_SNP.gff3` is the list of output SNPs (See details in Table 37 on page 193).

The file `<dibayes.output.prefix>_Consensus_Calls.txt` covers all positions that have coverage and provides general information about each position (See detail in Table 38 on page 194). Its flag column shows a list of codes of filters (See details in Table 39 on page 195) that the position fails to pass to be called as a SNP. It is very useful for identifying the reason of false negative SNP calls (the known SNPs that are not called by diBayes).

The file `<dibayes.output.prefix>_Consensus_Basespace2.fasta` is the updated fasta file of the chromosome sequence with SNP sites encoded in IUB codes and N for all non-covered positions.

The file `<dibayes.output.prefix>_quartiles.txt` lists the quartile and percentile information about the coverage and color quality value distribution of all positions of the chromosome (See example in Figure 57 on page 197).

In the root of output folder, the individual gff3 and fasta file in the chromosome subfolders are concatenated into the final gff3 and fasta file for the whole genome. Because of large sizes of individual Consensus_Calls files, we do not consolidate them into a summarized copy to save space.

Table 37 <diBayes.output.prefix>.gff3 file format description

Column name	Description	Example
##	Header comment lines	Input files and algorithm parameters.
#	Header of the results	—
seqid	The string ID of the sequence to which the start and end coordinates refer.	chr1
source	The source of the data.	SOLiD_diBayes
type	Sequence ontology derived type for this variation. For diBayes, this is always SNP.	SNP
start	Start position of the SNP.	420
end	End position of the SNP.	420
score	Calculated p-value of the SNP.	0.000000
strand	—	—
phase	—	—

Table 37 <diBayes.output.prefix>.gff3 file format description *(continued)*

Column name	Description	Example
Attributes:		
• genotype	Genotype in the form of IUB codes for bases observed in all the reads. **tigr.org/tdb/CMR/IUBcodes.html**	genotype=s
• reference	The base of the reference sequence at the current position.	reference=c
• coverage	The number of the reads that cover the current position.	coverage=52
• refAlleleCounts	The number of reads of the reference allele at the current position.	refAlleleCounts=22
• refAlleleStarts	The number of different start positions of reads having the reference allele at the current position.	refAlleleStarts=15
• refAlleleMeanQV	The mean of quality values of all reference allele reads at the current position.	refAlleleMeanQV=15
• novelAlleleCount	The number of reads of the most abundant non-reference allele at the current position.	novelAlleleCounts=24
• novelAlleleStarts	The number of different start positions of reads having the most abundant non-reference allele at the current position.	novelAlleleStarts=14
• novelAlleleMeanQ	The mean of quality values of all novel allele reads.	novelAlleleMeanQV=17
• diColor1	The most abundant allele in the reads (not necessarily the reference allele) in dicolor encoding (for example, 00, 01, ... 32, 33) - of 6 possible dicolors	diColor1=00
• diColor2	The second-most abundant allele in the reads.	diColor2=22
• Het	Heterozygosity flag 0=homozygous SNP 1=heterozygous SNP	het=1
• Flag	Filter summary flags for non-SNP positions. Always empty.	flag=

Table 38 <diBayes.output.prefix>_Consensus_Calls.txt file format description

File Name/Column	Description	Example
Chr	Chromosome/Contig number.	chr1
Position	Location of the SNP on the reference sequence.	442
Allele_DiColor1	The most abundant allele in the reads (not necessarily the reference allele) in dicolor encoding (for example 00, 01....32, 33) of 16 possible dicolors.	03
Allele_DiColor2	The second most abundant allele in the reads in dicolor encoding (for example 00, 01....32, 33) of 16 possible dicolors.	03
Reference	The base of the reference sequence at the current position.	C

Table 38 <diBayes.output.prefix>_Consensus_Calls.txt file format description *(continued)*

File Name/Column	Description	Example
Genotype	Genotype in the form of IUB codes for bases observed in all the reads.	C
P-value	Calculated p-value of the SNP.	1.00000
Flag	Flag indicating why a location was *not* called a SNP.	m4
Coverage	The number of the reads that cover the current position.	2
nCounts_1st_allele	The number of the most abundant allele at the current position.	2
nCounts_Reference _allele	The number of reads having the reference allele at the current position.	2
nCounts_NonReference _allele	The number of reads having the most abundant non-reference allele reads at the current position.	0
Ref-Avg-QV	The mean of quality values of all reference allele reads at the current position.	29
Novel-Avg-QV	The mean of quality values of all novel allele reads.	0
Heterozygous	Heterozygosity flag. Values are: 0 = homozygous SNP 1 = heterozygous SNP.	0
Algorithm	The algorithm used to call the current SNP. '-1': Not a SNP; bayes (Bayesian algorithm) quick (Frequentist algorithm).	-1 (Not a SNP)
Algorithm_Name	—	—

Table 39 <diBayes.output.prefix>_Consensus_Calls.txt flag column description

Flag	Filter meaning	Related key in *.ini file
Heterozygote		
h1	Insufficient coverage for heterozygous positions.	het.min.coverage.
h2	Not enough unique start positions.	het.min.start.pos
h3	Coverage is too high, filtered not a Het.	het.remove.high.coverage
h4	The fraction of the second-most common VALID color in the total of top two valid colors is higher than the threshold (usually a function of raw error squared).	het.min.allele.ratio
h5	Genome positions with sufficient coverage (20x) at which there is only 1 unique read position for all the reads. It could be a PCR error.	—
h6	The candidate SNP is evenly distributed over positions (not used).	—
h7	Novel allele is not on both strands (counting the reads.)	snp.both.strands
h8	Both alleles not evenly represented on both strands (doing statistical test on read distribution of the both strands.)	snp.both.strands

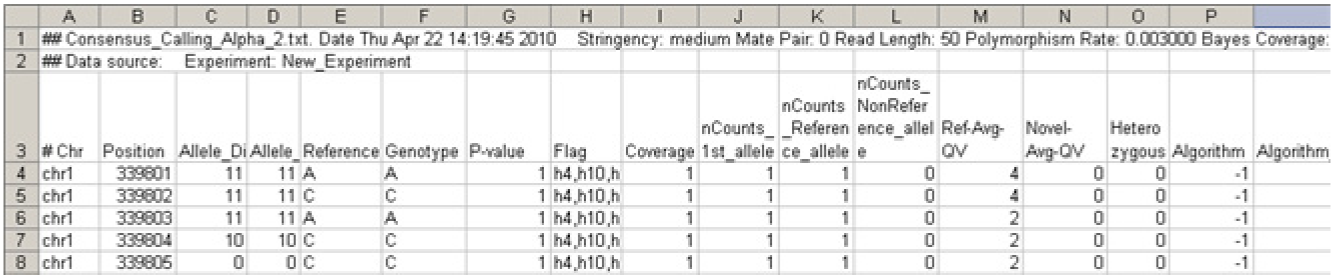
Table 39 <diBayes.output.prefix>_Consensus_Calls.txt flag column description *(continued)*

Flag	Filter meaning	Related key in *.ini file
h9	Second-most common base not more frequent than third	—
h10	There are no other valid SNPs, or the second-most common valid SNP (as a proportion of all valid SNPs) is less than half the 2-dibase error frequency	het.min.allele.ratio
h11	Sum of First and second-most common nucleotides must be at least this proportion, for example, 0.5, of all reads at this position.	het.min.ratio.validreads
h12	Reserved for future filters (not used).	—
h13	The quality value of the non-reference allele has much lower that of the reference allele.	—
h14	The quality value of the non-reference allele has much lower that of the reference allele. The non-reference allele has to be at d low frequency (rare variant).	—
h15	Genome position has low-quality value.	—
h16	Insufficient coverage of the reference allele.	—
h17	Insufficient coverage of the non-reference allele.	—
h18	Zero coverage.	—
h19	Insufficient number of start positions of non-reference allele.	—
h20	Insufficient coverage for either of two alleles, when neither allele is same as the reference.	—
h21	Quality value of non-reference allele too low (lower than a relative threshold depend on the distribution of all color quality values).	—
h22	Quality value of non-reference allele too low (lower than an absolute threshold).	het.min.nonref.color.qv
Homozygote		
m1	Insufficient coverage for a homozygous SNP.	hom.min.coverage
m2	Too many invalid dicolors at this position.	—
m3	Non-reference allele not on both strands.	snp.both.strands
m4	Insufficient coverage for a homozygous SNP.	—
m5	Second-most common allele too close in coverage to the first most common allele.	—
m6	Insufficient coverage (as a fraction of average coverage) for a homozygous call.	—
m7	Dicolor inconsistent with reference.	—
m8	Insufficient number of non-reference alleles.	hom.min.allele.count
m9	Allele ratio of second allele too high for homogyzous SNP.	—
m10	Insufficient number of start positions of non-reference alleles.	hom.min.start.pos
m11	No coverage.	—
m12	Quality value of non-reference allele too low (lower than an absolute threshold).	—

Table 39 <diBayes.output.prefix>_Consensus_Calls.txt flag column description *(continued)*

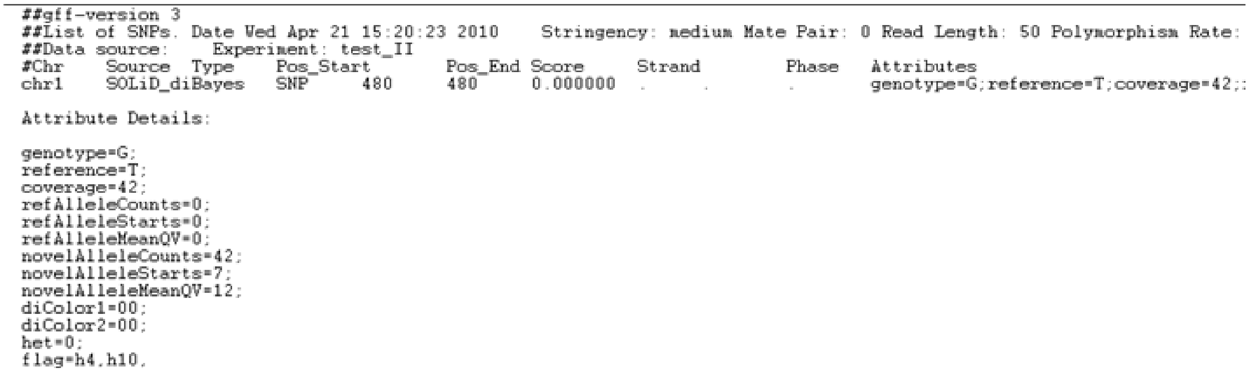
Flag	Filter meaning	Related key in *.ini file
m13	Quality value of non-reference allele too low (lower than a relative threshold depend on the distribution of all color quality values).	—

Output file examples



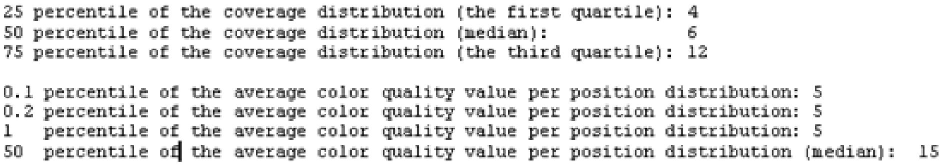
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	
1	## Consensus_Calling_Alpha_2.txt. Date Thu Apr 22 14:19:45 2010 Stringency: medium Mate Pair: 0 Read Length: 50 Polymorphism Rate: 0.003000 Bayes Coverage:																
2	## Data source: Experiment: New_Experiment																
3	# Chr	Position	Allele_Di	Allele_	Reference	Genotype	P-value	Flag	Coverage	nCounts_ 1st_allele	nCounts _Referen ce_allele	nCounts_ NonRefer ence_allel e	Ref-Avg- QV	Novel- Avg-QV	Hetero zygous	Algorithm	Algorithm
4	chr1	339801	11	11	A	A	1	h4,h10,h	1	1	1	0	4	0	0	-1	
5	chr1	339802	11	11	C	C	1	h4,h10,h	1	1	1	0	4	0	0	-1	
6	chr1	339803	11	11	A	A	1	h4,h10,h	1	1	1	0	2	0	0	-1	
7	chr1	339804	10	10	C	C	1	h4,h10,h	1	1	1	0	2	0	0	-1	
8	chr1	339805	0	0	C	C	1	h4,h10,h	1	1	1	0	2	0	0	-1	

Figure 55 <dibayes.output.prefix>_Consensus_Calls.txt file example



```
##gff-version 3
##List of SNPs. Date Wed Apr 21 15:20:23 2010    Stringency: medium Mate Pair: 0 Read Length: 50 Polymorphism Rate:
##Data source:     Experiment: test_II
#Chr     Source  Type     Pos_Start      Pos_End Score      Strand         Phase    Attributes
chr1     SOLiD_diBayes    SNP      480     480     0.000000   .       .         .      genotype=G;reference=T;coverage=42;:

Attribute Details:

genotype=G;
reference=T;
coverage=42;
refAlleleCounts=0;
refAlleleStarts=0;
refAlleleMeanQV=0;
novelAlleleCounts=42;
novelAlleleStarts=7;
novelAlleleMeanQV=12;
diColor1=00;
diColor2=00;
het=0;
flag=h4.h10.
```

Figure 56 <dibayes.output.prefix>.gff3 file example



```
25 percentile of the coverage distribution (the first quartile): 4
50 percentile of the coverage distribution (median):             6
75 percentile of the coverage distribution (the third quartile): 12

0.1 percentile of the average color quality value per position distribution: 5
0.2 percentile of the average color quality value per position distribution: 5
1   percentile of the average color quality value per position distribution: 5
50  percentile of the average color quality value per position distribution (median):  15
```

Figure 57 <dibayes.output.prefix>_quartiles.txt

FAQs – SNP finding using diBayes tool

1

SNP calling is taking too long. What can I do?

Parallelization is the best solution for running diBayes on large data sets. BioScope™ Software has implemented the parallel distribution of diBayes jobs for individual chromosomes. It can help the users to achieve the best running efficiency. In BioScope™ Software v1.2, diBayes uses *.bam files as its input and removes all large temporary files. The runtime for the diBayes is 30% faster than the runtime in previous versions of BioScope™ Software.

2

I seem to be finding too many SNPs — how do I troubleshoot?

Look at the properties of the SNPs and compare them to the properties of all the positions in the consensus_calls.txt file . Is the coverage of these SNPs much lower or much higher than average? Is the color quality value of the non-reference allele much lower than average? You might want to post-filter the results if you find these kinds of patterns. For example, you might want to remove SNPs with very low coverage, or very low color quality values, or p-values close to one. You might want to repeat the analysis with a more stringent setting for example, changing call.strengency from medium to high.

3

I seem to be missing SNPs — how do I troubleshoot?

First, try repeating the analysis with a lower stringency level (for example, changing call.strengency from medium to high). Look at the positions that you expect to be SNPs in the consensus_calls.txt file. Typically, you should find a list of flags that describe the reasons the position was not called as a SNP (see Table 37 on page 193). Look at the properties of this position. Visualizing the reads might be helpful here. Is the coverage much higher than average? The filter het.skip.high.coverage can remove heterozygous SNPs at positions of extremely high coverage because these have previously been observed to be variants in repeat regions rather than truly heterozygous SNPs at a single position. Are the reads strongly biased towards one strand, or is the non-reference allele missing from one strand? You probably want to perform a run with the at call.stringency=medium, or switch off the “both strands” requirement (snp.both.strand=0). Do all the reads have the same start position because of the way the sample was prepared? You need to reduce the number of unique start positions required to call a SNP (het.min.start.pos and hom.min.start.pos).

4

How can I control the sensitivity and specificity of SNP calling?

Different call stringency settings and filter settings may help users to achieve different sensitivity and specificity requirements. The more stringent the filters are, the less sensitivity the SNP call and the higher specificity in general. The parameters `reads.min.mapping.qv` (MQV) and `reads.min.alignlength.readlength.ratio` (ARR) are two filters that are very flexible for users' different needs. By changing one of them or both of them, users can filter out low confident reads and get SNP calls with high accuracy. The following figures show that changing MQV and ARR may affect the total number of SNP calls and the dbSNP concordance of the predictions.

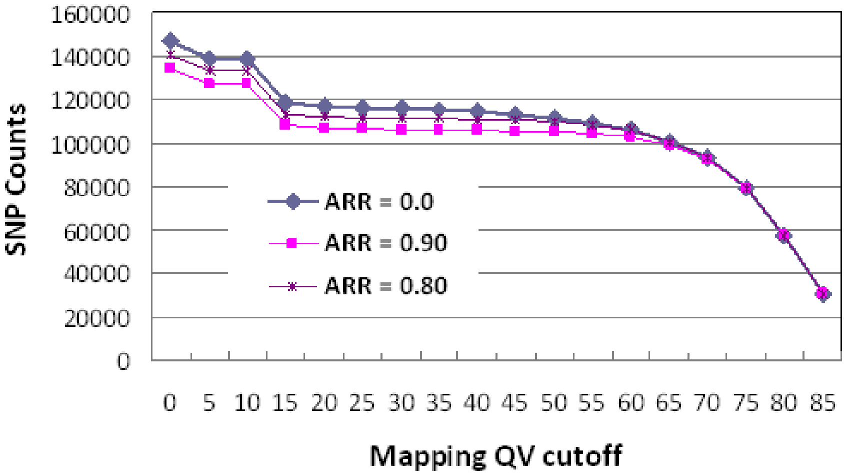


Figure 58 Total SNP calls on chromosome 1 of a HuRef long mate-pair data as a function of mapping qv cutoffs and ratios of alignment length and read length (ARR)

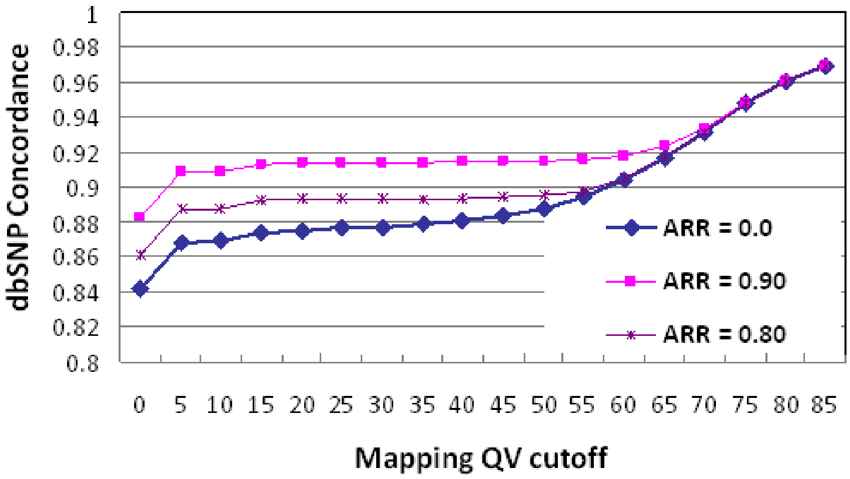


Figure 59 dbSNP concordance of SNP calls on chromosome 1 of a HuRef long mate-pair data as a function of mapping qv cutoffs and ratios of alignment length and read length (ARR)

5

Why does diBayes 4.0 do not support GFF file any more?

GFF files are text files, which take a lot of space. BAM files are binary files, which are usually one third of the sizes of their corresponding GFF files. Because of smaller physical size, BAM files can be read and write faster through the network within the framework of job distribution system of Bioscope^{TM} Software. Using BAM files significantly improves the speed of all tertiary tools. Furthermore, in the previous version of diBayes, we have to split GFF files and generate large temporary files to process different chromosomes. Using indexing of BAM files, diBayes can access any chromosome and location instantly. It can save a good amount of time by not splitting the files and by leaving no digital footprint. More importantly, Bioscope^{TM} Software v1.2 improved the mapping and pairing steps. Especially, a new informative mapping/paring quality value is introduced and assigned to every read. diBayes uses a threshold to filter low quality reads based on their mapping/pairing quality values to improve its SNP calling performance. This filter only works with the new mapping and pairing results coming in BAM format. Thus, diBayes only supports input files in BAM format.

6

What can I do with old GFF input files?

We recommend users to rerun the mapping and pairing with the raw reads through the new pipeline to generate BAM files. The new BAM files can be used on diBayes and all other tertiary applications.

12 Run the Find Human CNVs Tool

This chapter covers:

Human Copy Number Variation introduction

The Human Copy Number Variation (Human CNV) tool in Bioscope™ Software detects copy number variations in a data sample that is mapped to the human reference sequence hg18. The Human CNV tool currently supports only humans since normalization is species-specific.

Algorithm

The Human CNV tool algorithm has six steps:

- Preprocessing
- Coverage calculation
- Sampling into windows
- Normalization
- Segmentation
- Post processing

Preprocessing

User-defined configuration parameters in the cnv.ini file are validated and initialized. A working directory for intermediate files is created. The input *.cmap file is parsed, and a list of the names and lengths of the chromosome arms is loaded into memory.

Note: The CNV *.cmap file is specific to the CNV tool since it contains information about the location of the mappability files for each chromosome arm (used for normalization)

Coverage calculation

The coverage at every position, that is, the number of alignments spanning the position in the chromosome, is computed from the *.bam file. By default, all the alignments in the *.bam file are used to calculate coverage. Low quality alignments can be filtered out by modifying the mapping quality threshold in the cnv.ini file.

Optionally, the coverage computed for every chromosome in this step is output in *.wig file formats. The binary sizes for this coverage output can be defined by user in the cnv.ini file.

Sampling into windows

The algorithm divides the chromosomal region into windows of variable size, depending upon the mappability of the region. The window sizes are determined dynamically so that exactly the same number of mappable positions are in each window. The program distinguishes between mappable and unmappable positions using the precomputed mappability files.

The coverage mean for every window is computed by taking the average of the coverage values from all the mappable positions in each window. The log ratios between the coverage mean of every window and the expected coverage are computed. The mean of all windows in the whole chromosome arm is used as the expected coverage.

GC Correction

GC content is the number of G or C bases compared to the total number of bases in a particular region. In the regions of the genome where the percentage of GC content is either high or low, the coverage is observed to be decreased (see Figure 60 on page 203).

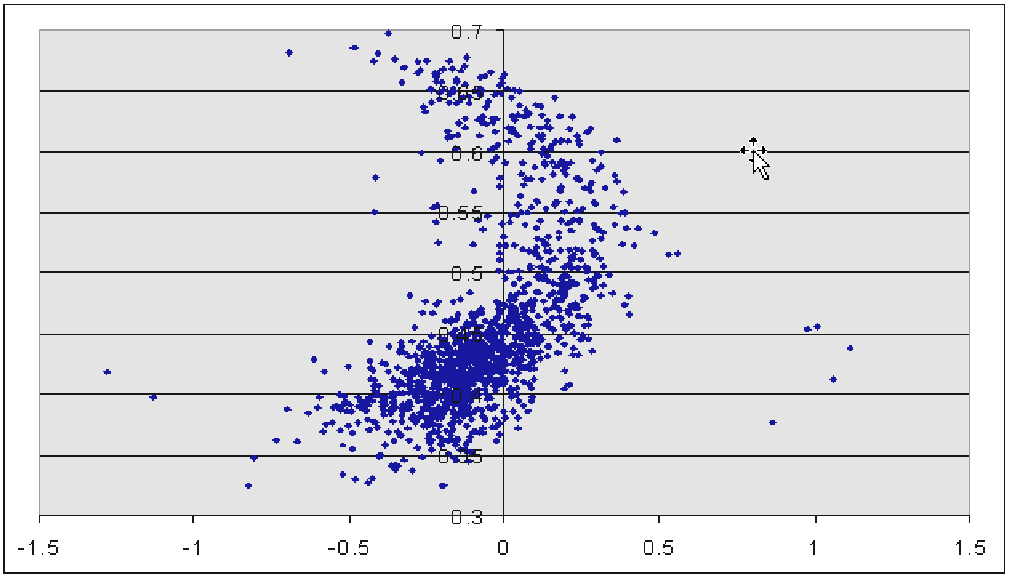


Figure 60 Extreme GC contact reduces coverage

Figure 60 shows the X -AXIS = log ratio (coverage of window/coverage of whole chromosome arm) and the Y- AXIS = Percentage of GC content in each window.

The algorithm normalizes this effect of GC contact by scaling the coverage of the windows with extreme GC content to the median coverage. The scaling factors for the windows are computed inline for every chromosome arm during runtime by the algorithm.

Normalization

The previous section explained how the log ratios computed in the previous step use the mean coverage of the local chromosomal arm as the expected value. However, if the algorithm is used with these settings, it cannot detect large CNV regions spanning more than half the length of the chromosome arm. To detect large CNVs, which can take whole chromosome arms or large portions of chromosome arms, the algorithm performs global normalization. The algorithm computes log ratios by using the median of the coverage means of all chromosome arms as the expected value (see Figure 61).

To skip the normalization step, set the local normalization parameter in the cnv.ini file to 1. By default, the algorithm performs global normalization if the number of valid chromosome arms provided for the analysis is nine or higher. In some cases, you might perform an initial run using Global Normalization, and then perform a run using Local Normalization to detect more fine-grained Human CNVs.

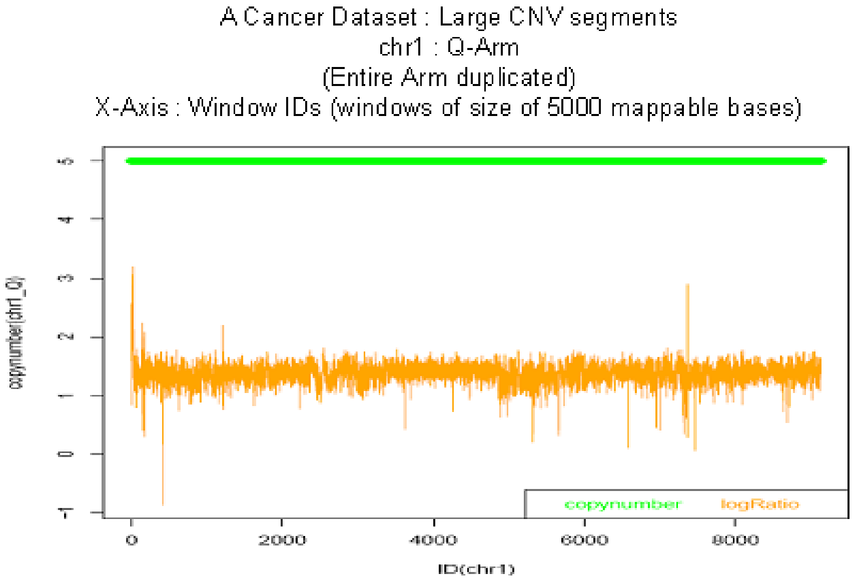


Figure 61 Large CNV segments example

Segmentation

The algorithm uses the Finite First Order Bayesian Hidden Markov Model (the model) to take the normalized log ratios of the windows as input and convert the continuous log ratio values into discrete copy number states (see Figure 62).

The model consists of ten states {0, 1, 2, 3, 4,…9}, where each state 'i' represents copy number "i". For diploid species, the state "2" is the normal state, states < 2 are copy number deletions, and states > 2 are copy number amplifications.

The prior probabilities and transition probability matrix for the model are trained using the Baum-Welch EM algorithm.

A copy number state with a p-value is assigned to each window using the Viterbi decoding algorithm. The p-value is a measurement of statistical significance. In general, the lower the p-value, the more likely it is that the prediction will be true.

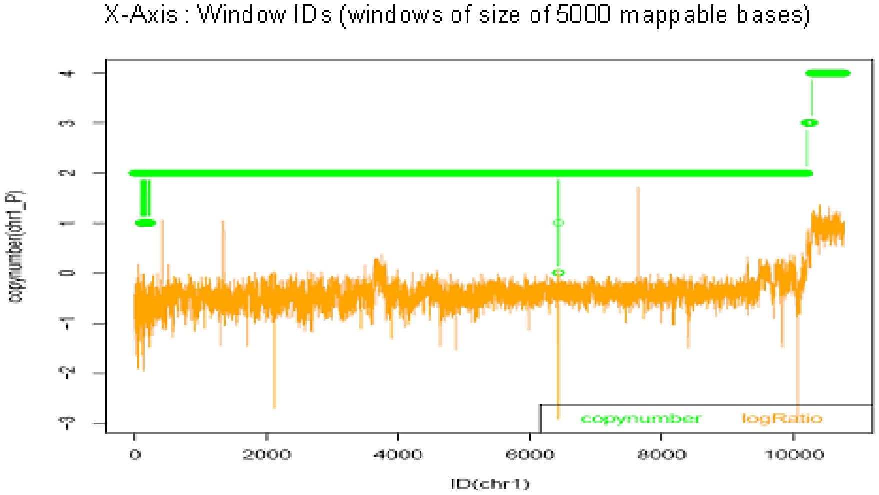


Figure 62 Segmentation example

Post processing

Neighboring windows are merged into a segment when each window has:

- The same copy number.
- Copy number deletions.
- Similar copy number amplifications. Amplifications are copy numbers greater than two and copy numbers that differ by one.

The algorithm applies filtering criteria on the Human CNV calls according to parameters defined in the cnv.ini file. The filtering criteria can include parameters defined for minimum mappability, number of continuous blocks with copy number greater than two, and so forth. The algorithm uses the *.gff format for structural variants to format the Human CNV segments that pass all of the filtering criteria. See Table 42 on page 215 for a description of the cnv*.gff file format.

cnv.ini file example

The following section shows a typical example of the cnv.ini file. For a description of the cnv.ini file parameters, see Table 40 on page 207.

IMPORTANT! Each time you run the Find Human CNVs tool, whether you use the command line or the web interface, you must update the output prefix, define the full path to the *.cmap file, and define the full path to at least one *.bam file.

IMPORTANT! Before you begin a run, you must verify the settings for each parameter that is highlighted in **bold** in the *.ini file example shown in the next section.

```
####################################
####################################
##
##  global parameters
##
import ../globals/global.ini
reference = ${reference.dir}/human_var/chr20.validated.fasta


##*****************************************************
##  CNV tool
##****************************************************
# mandatory parameters
# --------------------

# Parameter specifies whether to run or not cnv pipeline. [1: to
run, 0:to not run]
cnv.run = 1

# CNV Output Prefix
cnv.output.prefix = exampleExperment

# Comma-separated paths to Input BAM files
coverage.file.info = ${output.dir}/pairing/F3-F5-P2-Paired.bam
```

```
# Format of the coverage files provided (GFF|BAM)
coverage.format = BAM

# Absolute path of the CMAP file
cmap = ${base.dir}/referenceMapping_pe.cmap

# Path to the output directory
cnv.output.dir = ${output.dir}/cnv

# Path to the log directory
cnv.log.dir = ${log.dir}/cnv

# Path to the intermediate directory
cnv.intermediate.dir = ${base.dir}/intermediate

# optional Parameters
# ---------------------

# Window Size - Size of the Window Block to be considered as a
region
#window.size=5000

# Trim Distance - Distance in Kilo bases to be trimmed from the
extreme ends of the chromosome arms.
#trim.distance = 1000

# Normalization - Global or Local Normalization Global - 0 Local
-1
#local.normalization = 0

# Gender - Gender Female 1 Male 2
#gender = 2

# CNV Min Quality
#cnv.min.quality = 0

# Max pVal - Maximum p-value of the CNV segment
#max.pval = 1.0

# UMinMap - Minimum mappability percentage for the regions to be
shown as having copy number less than 2
#unimap = 10

# OminMap - Minimum mappability percentage for the regions to be
shown as having copy number greater than 2
#ominmap = 10

# UminBlocks - Minimum number of continuous Blocks with copy
number less than 2
#uminblocks = 2

# OMinBlocks - Minimum number of continuous blocks with copy
number greater than 2
#ominblocks = 2
```

```
# Max log Ratio - Maximum Log Ratio Threshold for Copy Number
Deletion Regions
#max.log.ratio = -0.678

# Min Log Ratio - Minimum Log Ratio Threshold for Copy Number
Amplification Regions
#min.log.ratio = 0.375

# Write Coverage - Write coverage
#write.coverage = 0
```

cnv.ini file parameter description

Table 40 cnv.ini file parameter description

Parameter name	Default value	Description
Mandatory parameters		
cnv.run	1	Specifies whether or not to run the Human CNV tool. Enter 0 if you do not want to run the Human CNV tool.
cnv.output.prefix	—	The name of the experiment.
coverage.file.info	—	The comma-separated paths to the input BAM|GFF files.
coverage.format	BAM	The format of the coverage files provided [GFF | BAM]
cmap	—	The absolute path to the *.cmap file.
cnv.output.dir	—	The path to the output directory.
cnv.log.dir	—	The path to the log directory.
cnv.intermediate.dir	—	The path to the intermediate file directory.
Optional parameters		
window.size	5000	The size of the window block to be considered as a region.
trim.distance	1000	The distance in kilobases to be trimmed from the extreme ends of the chromosome arms.
local.normalization	0	Whether global or local normalization should be carried out. Enter 0 for global normalization. Enter 1 for local normalization.
gender	2	The gender of the human data source. Enter 1 for female or 2 for male.
cnv.min.quality	0	The minimum quality value.
max.pval	1.0	The maximum p-value of the Human CNV segment.
unimap	10	The minimum mappability percentage for the regions to be shown as having a copy number < 2.
ominmap	10	The minimum mappability percentage for the regions to be shown as having copy number > 2.
uminblocks	2	The minimum number of continuous blocks with copy number < 2.
ominblocks	2	The minimum number of continuous blocks with copy number > 2.
max.log.ratio	-0.678	The maximum log ratio threshold for copy number deletion regions.

Table 40 cnv.ini file parameter description *(continued)*

Parameter name	Default value	Description
min.log.ratio	0.375	The maximum log ratio threshold for copy number amplification regions.
coverage.wsize	1000	Size of the bin to write coverage output. The "bin" is the size of the window block to be considered as a region for writing coverage output. Mean coverage of all bases in each of these windows will be output.
write.coverage	0	Whether coverage files should be output or not. Enter 1 to write coverage output in *.wig format. Keep the default value if you do not want to output coverage files.

Prepare to run the Find Human CNVs tool

By default, the tool does not call Human CNVs that are within 1 MBase of the centromeres and telomeres of the chromosomes. A centromere is a region of DNA typically found near the middle of a chromosome where two identical sister chromatids come in contact. The centromere is involved in cell division as the point of mitotic spindle. A telomere is a region of repetitive DNA at the end of a chromosome. The telomere protects the end of the chromosome from deterioration. The distance from the centromeres and telomeres in which Human CNVs are not called is a parameter that can be modified in the cnv.ini file.

Select the required input files

The required input files are available only for the human reference sequence hg18. BioScope™ Software includes the human reference sequence hg18. The reads should have been mapped to human reference sequence hg18.

Before you can run the Find Human CNVs tool you must know:

- The absolute path to at least one *.bam file.
- The absolute path to the predicted mappability files.
- The absolute path to the *.cmap file.

Complete the prerequisites

1. Download `hs_CNV_data_<version>.tar.gz` from **solidsoftwaretools.com** (see "hs_CNV_data file download and installation"

2. Complete the applicable prerequisites described in Chapter 3, "Before you Begin" on page 35.

3. Login to the BioScope™ Software cluster. Change to the working directory and update the cnv.ini file with information that applies to the Human CNV run. See "cnv.ini file example" on page 205.

4. Create an output prefix. Output prefixes are case-sensitive. Use an underscore to separate terms in an experiment name. Example: Experiment_1.

5. Complete the resequencing mapping/pairing process on the primary data from the instrument.

hs_CNV_data file download and installation

The hs_CNV_data folder contains data that is required to perform a Human CNV run. You can unzip the folder in any directory that you choose. The folder contains:

- Predicted mappability files for human reference sequence hg18.
- A set of hg18 reference files split per chromosome in *.fasta file format.
- A cnv.cmap file.

You must provide the absolute path to the folder in the cnv.cmap file. Be sure that you provide the path to the cnv.cmap file location (absolute path only) has to be updated in the above mentioned cmap file and that cmap file should be provided as *.cmap input to the Human CNV tool. Be sure that you update the cnv.ini file with the path to the *.cmap file.

Run the Find Human CNVs tool from the command line

Although several different software programs are involved in the run, a single command generates all of the related programs required to complete the run. The *.plan file that is specified in the command syntax controls the order in which BioScope™ Software runs the related programs.

Start the run

1. Connect to the BioScope™ Software cluster and login with a user ID that has write privileges on all of the directories that BioScope™ Software uses when the tool runs.
2. At a command prompt, enter:
   ```
   bioscope.sh -l filename.log filename.plan
   ```

Do not log out of the BioScope™ Software cluster.

Check the run status from the command line

1. Navigate to the log directory that is defined in the cnv.ini file. For example, you might enter:
   ```
   cd /data/results/tertiary/cnv/log
   ```
2. Open `bioscope.`*`yyyymmddhhmmss`*`.log`.
3. Scroll to the end of the file.

The run is complete if you see an entry similar to:

```
15 Apr 2010 03:16:32,537  INFO [main] PluginJobManager:130 -
>>>> END of PluginJobManager >>>> date DURATION=4 minutes 33
secs

15 Apr 2010 03:16:32,537  INFO [main] EventTransportFactory:129
- Closing JMS connection and session
```

Run the Find Human CNVs tool from the web interface

The instructions in this section assume the following system conditions:

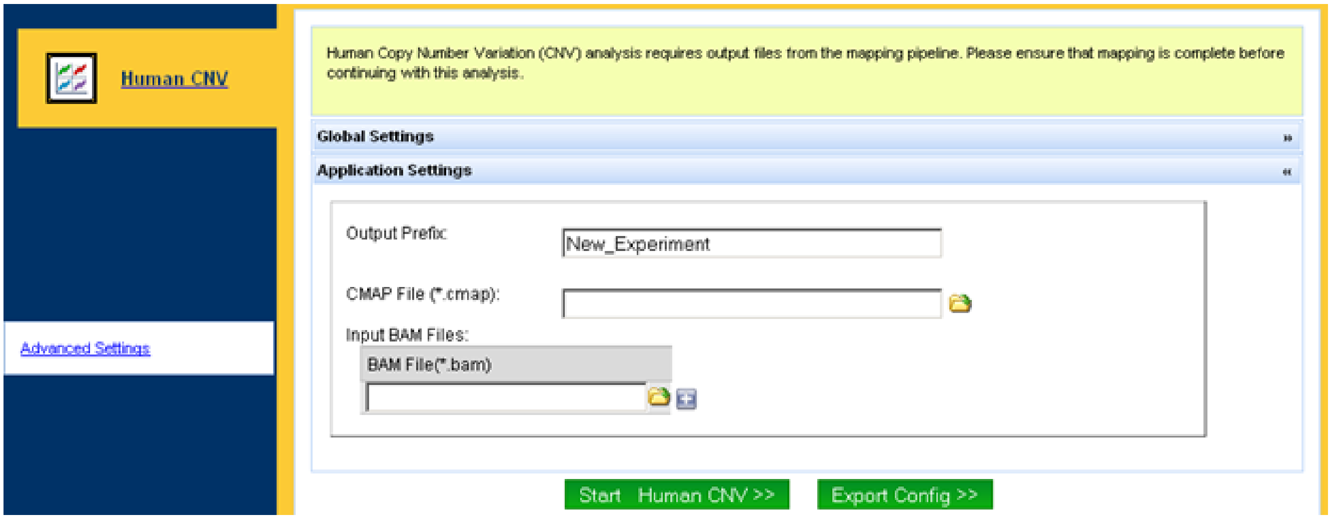
- The Java Messenger, Tomcat, and Apache services are running on the BioScope™ Software cluster.
- You are using Internet Explorer versions 6 or 7, or Mozilla 3.0.1.

- You have planned the name of the CNV output prefix.
- Mapping and pairing is complete.

1. Launch a browser and enter the BioScope™ Software URL: http://<hostname>:8080/bioscope
2. Click **Find Human CNVs**.

The Find Human CNVs page has two windows and one link (see Figure 63):

- Global Settings
- Applications Settings
- Advanced Settings



Human Copy Number Variation (CNV) analysis requires output files from the mapping pipeline. Please ensure that mapping is complete before continuing with this analysis.

Output Prefix: New_Experiment

CMAP File (*.cmap):

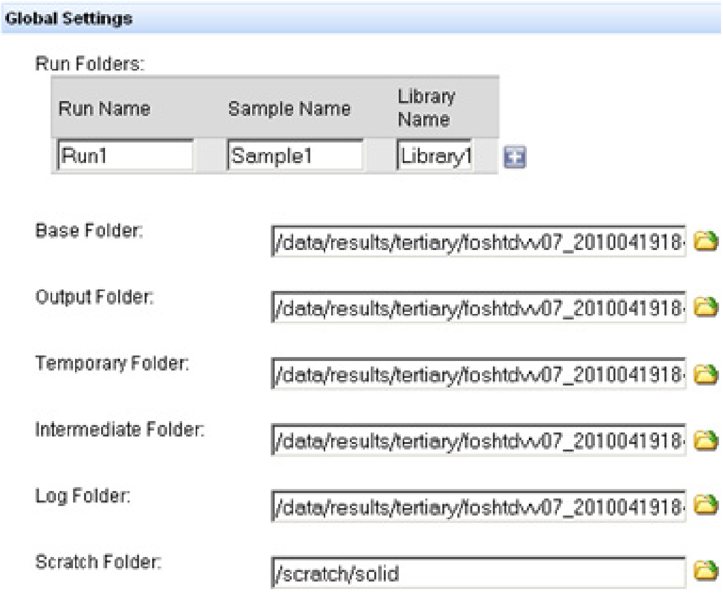
Input BAM Files:

BAM File(*.bam)

Figure 63 Find Human CNVs web page example

Global Settings description

The Global Settings window displays the default values for the folders that Bioscope™ Software creates for the files resulting from the Find Human CNVs run. The window also has fields where you can update default values for the Run Name, Sample Name, and Library Name of the primary data that was exported to Bioscope™ Software from the instrument. Figure 64 on page 211 shows an example of the Global Settings window.



Global Settings

Run Folders:

Run Name	Sample Name	Library Name
Run1	Sample1	Library1

Base Folder: /data/results/tertiary/foshtdw07_2010041918

Output Folder: /data/results/tertiary/foshtdw07_2010041918

Temporary Folder: /data/results/tertiary/foshtdw07_2010041918

Intermediate Folder: /data/results/tertiary/foshtdw07_2010041918

Log Folder: /data/results/tertiary/foshtdw07_2010041918

Scratch Folder: /scratch/solid

Figure 64 Find Human CNVs Global Settings section example

Customize the default folder structure (optional)

The folders store the results files generated by each Find Human CNVs run. Bioscope™ Software automatically creates the default folder structure (below) for each Find Human CNVs run: `/data/results/tertiary/` *`headnode_yyyymmddhhmmss_x`*

Complete the following steps to change the default directory structure:

1. Click 📂 in the Base Folder field. The File Browser dialog appears.
2. In the **Look in** field, type a custom directory path, for example, `/home/data`
3. Click **Open**.
4. The folders reflect the updated directory structure.

Note: If you change the default directory structure, the Output, Temporary, Intermediate, and Log folders become subdirectories of the Base Folder.

Update the Run Folder settings (optional)

You can accept the default values in the Run Name, Sample Name and Library Name fields. In this context, "run" refers to the primary data that was exported to Bioscope™ Software from the instrument.

To change the default values for the Run Folders:

1. Enter the updated run name in the Run Name field.
2. Enter the updated sample name in the Sample Name field.
3. Enter the updated library name in the Library Name field.

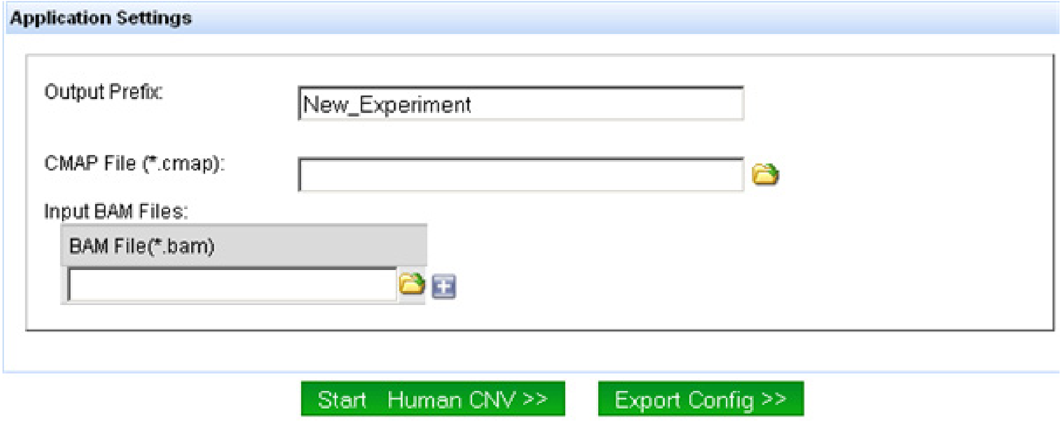
4. Optional: Click ⊞ to add a row for a second run folder.
5. Optional: Enter a Run Name, a Sample Name and a Library Name in the new row.

Advanced Settings description

Click **Advanced Settings** to view the current default values defined by Bioscope™ Software for the Find Human CNV tool. Do *not* change any Advanced Settings unless instructed to by the Bioscope™ Software administrator.

Application Settings description

In the Application Settings section (see Figure 65 on page 212), you update the Output Prefix and define the absolute paths to the *.cmap and input *.bam files. You must update those three parameters each time that you run the Find Human CNVs tool. You also start the Find Human CNV run from the Applications Setting section. The Export Config >> button is only used with the tool that processes barcoded libraries (see Appendix C, "Batch Analysis of Barcoded Library Data" on page 319).



Application Settings

Output Prefix: New_Experiment

CMAP File (*.cmap):

Input BAM Files:

BAM File(*.bam)

Figure 65 Find Human CNVs Application Settings window

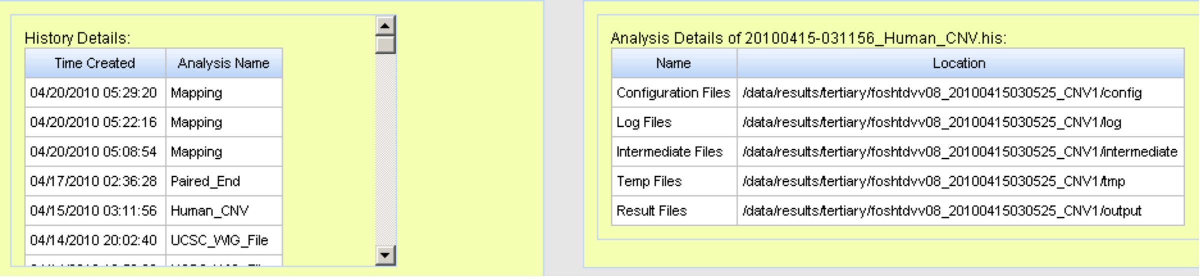
Start the Find Human CNV tool run

1. Define the Output Prefix or accept the default name.
2. Click 📂 in the CMAP File (*.cmap) field. The File Browser window appears.
3. Define the directory path to the *.cmap file.
4. Click **Open**.
5. Click 📂 in the BAM File(*.bam) field. The File Browser window appears.
6. Define the directory path to the *.bam file.
7. Click **Open**.
8. **Optional:** Click ⊞ to include additional *.bam files. Repeat steps 6 and 7.
9. Click Start Human CNV >> to start the run.
10. At the job submission dialog, click **OK** after you have verified the folder locations.

Check the status of the run from the web interface

1. Click **History**. The History window appears. is displayed in the left pane. The History Details table shows the Time Created and Analysis Name for all runs performed on the BioScope™ Software cluster.

2. Scroll the History Details table and select a Human_CNV run, based on the data in the Time Created column (see Figure 66).



History Details:

Time Created	Analysis Name
04/20/2010 05:29:20	Mapping
04/20/2010 05:22:16	Mapping
04/20/2010 05:08:54	Mapping
04/17/2010 02:36:28	Paired_End
04/15/2010 03:11:56	Human_CNV
04/14/2010 20:02:40	UCSC_WIG_File

Analysis Details of 20100415-031156_Human_CNV.his:

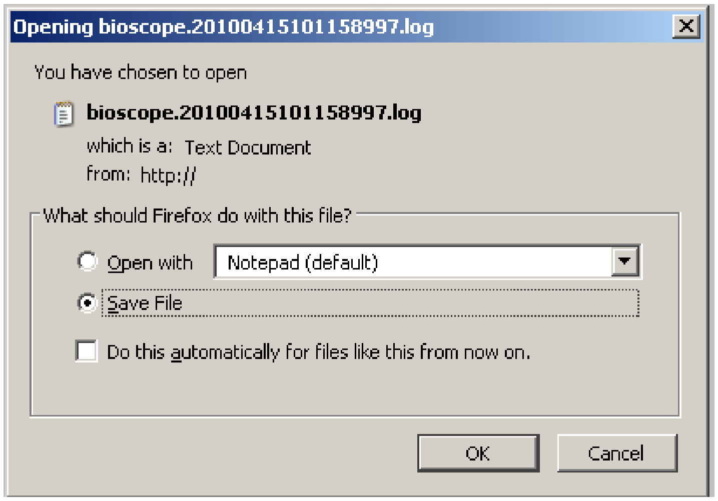
Name	Location
Configuration Files	/data/results/tertiary/foshtdvv08_20100415030525_CNV1/config
Log Files	/data/results/tertiary/foshtdvv08_20100415030525_CNV1/log
Intermediate Files	/data/results/tertiary/foshtdvv08_20100415030525_CNV1/intermediate
Temp Files	/data/results/tertiary/foshtdvv08_20100415030525_CNV1/tmp
Result Files	/data/results/tertiary/foshtdvv08_20100415030525_CNV1/output

Figure 66 History details and analysis details for a Find Human CNVs tool run

3. Double-click the Log Files row in the Analysis Details table. The File Browser dialog opens. Click **Resend** if your browser displays a message.

4. Select the `bioscope.`*`yyyymmddhhmmss`*`.log` file.

5. Click **Download**.
   - Click **Open with** and click **OK** to view the log file in Notepad or select a different text editor.
   - Click **Save File** to copy the file to your workstation.



Opening bioscope.20100415101158997.log

You have chosen to open

bioscope.20100415101158997.log

which is a: Text Document

from: http://

What should Firefox do with this file?

○ Open with Notepad (default)

● Save File

□ Do this automatically for files like this from now on.

Figure 67 Log file download page example

6. Scroll to the end of the file.

The run is complete if you see an entry similar to:

```
15 Apr 2010 03:16:32,537  INFO [main] PluginJobManager:130 -
>>>> END of PluginJobManager >>>> date DURATION=4 minutes 33
secs
```

```
15 Apr 2010 03:16:32,537  INFO [main] EventTransportFactory:129
- Closing JMS connection and session
```

Find Human CNVs results file format description

This section describes the file format of the *.out files and the *.gff file created by the Find Human CNVs tool run.

*.out files

The*.out files are:

- <Experiment_Name>_AllSegments.out
- <Experiment_Name>_CNVs.out
- <Experiment_Name>_CNVs_unfiltered.out

where <Experiment_Name> is the value defined for the Output Prefix.

The three files share a common file format. The formats are described in Table 41. You can view the files in a text editor or a spreadsheet application (see an example in Figure on page 214).

*.gff file

The tool creates one <Experiment_Name>.gff.file. The file formats are described in Table 42. You can view the <Experiment_Name>.gff. file in a text editor or a spreadsheet application (see an example in Figure 69 on page 216). You can also visualize the file in a browser such as the Integrative Genomics Viewer (IGV), or in a browser such as the UC Santa Cruz genome browser, which is available from UC Santa Cruz. You can download the IGV browser from the Broad Institute Web site. For more information, go to **www.broadinstitute.org/igv,** or **genome.ucsc.edu/**

Table 41 <Experiment_Name>_*.out file format descriptions

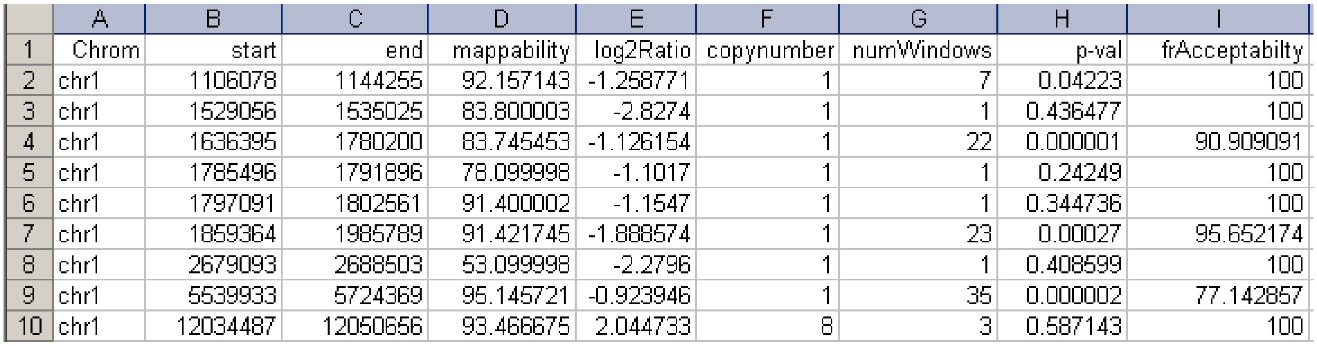
Column Title	Description	Example
Chrom	Chromosome number.	chr1
start	Start location of the CNV region	1636395
end	End location of the CNV region	1780200
mappability	Fraction of mappable bases in the CNV region	83.745453
log2Ratio	Mean of Log2Ratios of the windows in the CNV region	-1.126154
copy number	Copy number of the region	The copy number is relative to a diploid genome, with a normal copy number of 2.
numWindows	Number of windows in the CNV region	22
p-val	p-value of the CNV call for the region	0.00001 is very confident: 0.99 is not at all confident.
frAcceptability	Fraction windows in the Region that passed all the filtering criteria individually. Filtering criteria includes minimum mappability, minimum number of windows, min log ratio, max log ratio and max p-value.	90.909091

Table 42 <Experiment_name>_CNVs.gff file format

Column Title	Description	Example
seqid	The ID of the sequence to which the start and end coordinates refer.	chr1
source	Free text-qualifier indicating the algorithm or method that generated the feature.	AB_CNV_PIPELINE
type	Sequence ontology derived type for this variation.	repeat_region
start	Start position of the CNV Region.	1144255
end	End position of the CNV Region.	0.04223
score	p-Value of the CNV Region.	—
strand	Not used for this output.	—
phase [attribute]	Not used for this output.	—
attributes		
copynumber	Copy number of the region.	1
log2Ratio	Mean of Log2Ratios of the windows in the Human CNV region.	-1.258771
numWindows	Number of windows in the Human CNV region.	23. Usually, the larger this number is, the more confident the CNV call.
mappability	Fraction of mappable bases in the Human CNV region.	91.421745. The maximum value is 100. A low number may indicate that this region is difficult to map and so may have an increased likelihood of being a false positive CNV call.

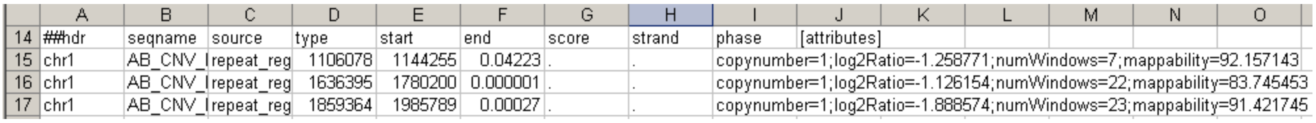
Find Human CNVs results file examples

This section provides examples of the *.out files and the *.gff file created when you run the Find Human CNVs tool.



	A	B	C	D	E	F	G	H	I
1	Chrom	start	end	mappability	log2Ratio	copynumber	numWindows	p-val	frAcceptabilty
2	chr1	1106078	1144255	92.157143	-1.258771	1	7	0.04223	100
3	chr1	1529056	1535025	83.800003	-2.8274	1	1	0.436477	100
4	chr1	1636395	1780200	83.745453	-1.126154	1	22	0.000001	90.909091
5	chr1	1785496	1791896	78.099998	-1.1017	1	1	0.24249	100
6	chr1	1797091	1802561	91.400002	-1.1547	1	1	0.344736	100
7	chr1	1859364	1985789	91.421745	-1.888574	1	23	0.00027	95.652174
8	chr1	2679093	2688503	53.099998	-2.2796	1	1	0.408599	100
9	chr1	5539933	5724369	95.145721	-0.923946	1	35	0.000002	77.142857
10	chr1	12034487	12050656	93.466675	2.044733	8	3	0.587143	100

Figure 68 <Experiment_Name>_*.out file format example



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
14	##hdr	seqname	source	type	start	end	score	strand	phase	[attributes]					
15	chr1	AB_CNV_I	repeat_reg	1106078	1144255	0.04223	.	.	copynumber=1;log2Ratio=-1.258771;numWindows=7;mappability=92.157143						
16	chr1	AB_CNV_I	repeat_reg	1636395	1780200	0.000001	.	.	copynumber=1;log2Ratio=-1.126154;numWindows=22;mappability=83.745453						
17	chr1	AB_CNV_I	repeat_reg	1859364	1985789	0.00027	.	.	copynumber=1;log2Ratio=-1.888574;numWindows=23;mappability=91.421745						

Figure 69 <Experiment_Name>.gff file format example

FAQs – Find Human CNVs

1

Does the Find Human CNVs tool work on any species?

No. The current version of the tool works only for the Human hg18 reference. The tool cannot work on other species because Bioscope™ Software does not have predicted mappability files for any species other than human.

2

How do various parameters affect the sensitivity and specificity of the CNV calls?

The default configuration are designed such that CNV calls are made with balance between sensitivity and specificity.

If users wish to increase the sensitivity, they can increase the "window.size" value, use only high quality alignments by filtering out the low quality ones using "cnv.min.quality" and by making the filtering parameters like max.pval, uminmap, ominmap, uminblocks, ominblocks to be more restrictive.

If users wish to increase the specificity, they can work at higher resolution by using smaller "window.size", by using lower value for cnv.min.quality and by making the filtering parameters less conservative.

3

What do the mappability files contain?

The mappability files are essentially a representation of 25 mer fragment 50 mer Fragment, or 50 mer mate-pair files from the reference that are mapped back to the reference. The mappability files have one row per every position, indicating whether that position is or is not uniquely mappable. A *mer* is the number of the bases per read.

4

Which different sets of mappabilty files are provided? How does the tool decide which set of mappability files to use?

Bioscope™ Software provide three sets of mappability files:

- 25.2 Fragment
- 50.4 Fragment
- 50X2 Mate-pair

The tool automatically selects the appropriate set of files, based on the following criteria (see Table 43):

Table 43 Mappability file description

Mappability files	Data type	Read length	Insert size
25 mer fragment files	mate-pair, fragment	<50	Any.
50 mer fragment files	mate-pair, fragment	>=50	Any.
50X2 mer mate-pair files	—	—	—

4

Can users generate mappability files if they have *.fasta files?

Bioscope™ Software does not provide any applications that allows users to generate their own customized mappability files. Contact your Life Technologies BioInformatics FAS for additional questions.

5

Can users perform a Human CNV analysis on other species if they have mappability files for that species?

As well as a mappability file, a CNV *.cmap file with valid chromosome ranges (including the start and end of the p-arm and q-arm and the location of the mappability files for each chromosome arm) must be provided.

6

Does the tool require the PBS cluster to run?

No. The PBS cluster is not required for the run.

7

When should we change the default value of "-window-size"?

The size of Human CNV segments detected by the tool is directly dependent on the value of "-window-size". Typically, the smallest Human CNV segment that can be detected is at least twice the value of the "-window-size" size. The smaller the window size, the smaller the CNV that can be detected. However very small CNVs may be more likely to be false positives (or at least, under-represented in existing public databases).

As the "-window-size" size decreases, the time taken for the Human CNV analysis and the sizes of the files generated increases significantly.

8

When should we use the "-local-normalization"?

To detect smaller Human CNVs (kB scale) with tumor samples, normalize by the local chromosome context to detect these smaller Human CNVs. If you do not use “-local-normalization”, large perturbations in ploidy across the whole genome might confuse detection. Ploidy is the number of complete sets of chromosomes in a biological cell. In humans, the somatic cells that compose the body are diploid (containing two complete sets of chromosomes, one set derived from each parent).

The tool applies various user-configurable filtering criteria, such as minimum mappability, number of continuous blocks, and so forth on the Human CNV calls.

9

How are the p-values for Human CNV segments computed?

The p-values for Human CNV segments are computed using probabilities from the Finite First Order Bayesian Hidden Markov Model (the model). Sequence of log ratios of coverage per window are given as input to the model for segmentation. The tool calculates the most probable Human CNV-state, that is, the hidden state, for each window using "Forward-Backward" Algorithm on the model. For example, if window 'W' is assigned Human CNV state 'c' with a probability 'p', then the p-value of that window is '1-p'. The calculations control Human CNV states and p-values for all windows. In the next step, the tool algorithm merges the neighboring windows with similar copy number states into a Human CNV segment. The p-value of that Human CNV segment is given by the minimum p-value of all merged windows.

13 Run the Find Inversions Tool

This chapter covers:

Inversion algorithm overview

The Inversion Tool exploits the large insert size of SOLiD™ mate-pair libraries to detect important, but often poorly-characterized, inversion polymorphisms. The large insert size is critical for spanning the repeat regions that are associated with inversion break points.

In its simplest form, the algorithm collects evidence for an inversion by observing accumulations of pairs with correct relative positioning, but with an improper orientation. For SOLiD™ mate-pair libraries, this requires tags to be mapped to opposite strands. For example, if the R3 tag is to the left of an F3 tag, but the R3 tag maps to the top strand and the F3 tag maps to the bottom strand, this pair provides inversion evidence, specifically that the R3 tag is to the left of the inversion starting break point and that the F3 tag is to the right of the break point. Once the evidence is collected, candidate inversion break points are scored, paired, and ranked. Additional evidence is provided by a scan for drops in the coverage by normal mate-pairs (see Figure 70).

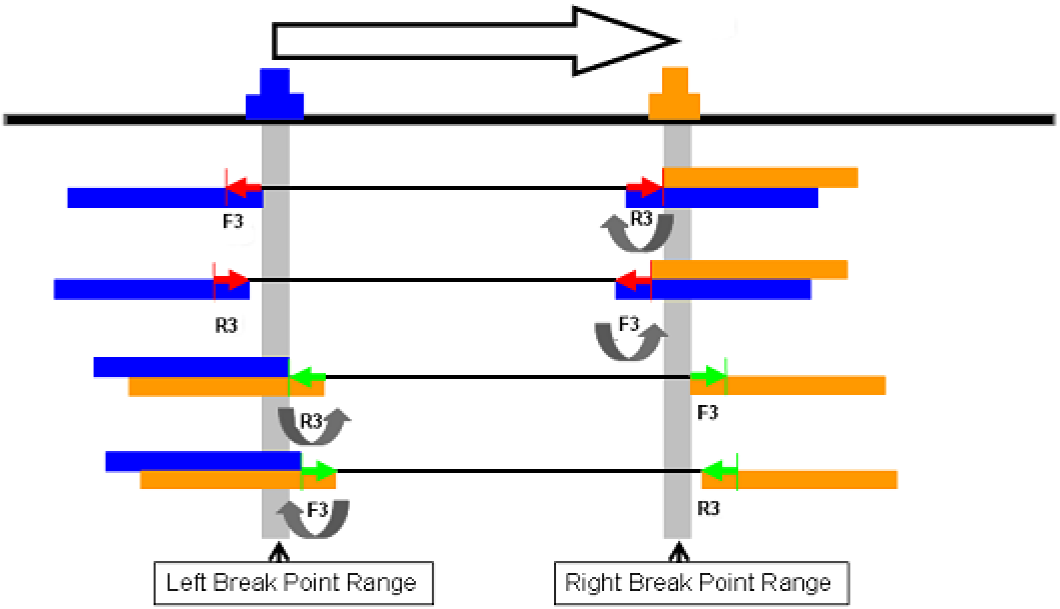


Figure 70 Inversions and break point ranges

Figure 70 depicts the following elements:

Thick black line	The sequenced genome.
Thin black lines	Mate-pairs.
Red and green arrows	Two ends of an inverted mate-pair.
Blue and orange bars	The maximum distance separating the two ends of a normal mate-pair (AAA mates).

By definition, an inversion has two breakpoints: a starting breakpoint and an ending breakpoint. The Inversion Tool plug-in uses SOLiD™ mate files or mate-pair BAM files as input (using paired-end BAM files is not recommended due to their small insert size). The number of mate-pairs supporting an occurrence (of both starting and ending inversion breakpoints) are counted for each base pair as breakpoint scores. Candidate breakpoint ranges are genomic ranges corresponding to local peaks of counts above a predetermined score threshold. The clone insert size constraints specify a minimum and a maximum (blue and orange horizontal bars) distance separating the two ends (small green or red arrows) of a mate-pair (thin black lines) in the sequenced genome (thick black line). Each green mate-pair suggests a starting breakpoint of an inversion occurring to the left, and an ending breakpoint between its two tags. The green pairs then define the range of the starting breakpoint by contributing positive counts to all base pairs to the left of their left tags within 1 max. They also help refine the ending breakpoint range by contributing negative counts to all base pairs 1 max to the left and 1 max to the right. The red mate-pairs contribute counts in a similar fashion.

When small inversion detection is enabled (with the `recover.tiny.inversions` parameter key), a second pass of the algorithm is performed, but with a smaller window for the scoring function, allowing the detection of inversions of 200 base pairs, or even lower, depending on coverage.

Like other SOLiD™ tools, the inversion tool outputs results in GFF format for easy viewing in common genome browsers (for example, the SOLiD™ alignment browser). Because of the inherently ambiguous nature of the detection, the start and end locations represent the widest possible range. More details are provided by the GFF attributes including the break point ranges and scores.

While the same algorithm can be applied to SOLiD™ paired-end libraries, in practice, the smaller insert size makes inversions more difficult to detect. Without the large insert size of mate-pair libraries, both tags would place into repeat regions that typically flank inversions, making tag placement ambiguous.

Inversion algorithm details

Input data

The first step in the detection of inversions is the collection of inverted mates. These are filtered from the BAM file by selecting for mates of opposite orientation. In pairing category terms, this includes BA*, BB*, and AB*. These records are written to the intermediate directory (`inversion.intermediate.dir`) so that the filtered set can be reused. It is important that either the library.type parameter be set or that the LB field be properly constructed to indicate a mate-pair library.

In previous versions (prior to BioScope™ Software v1.2), unique or non-redundant mates files were used as input to the Inversion Tool. Non-redundant mates are selected from BAM file input using the PCR duplicates flag (`0x0400`). Pairing quality values can be used to select unique records via the `inversion.min.qv` parameter key. Values greater than 20 should be unique, but criteria may be different for different applications.

The Pairing "SV" output filter is specifically designed to capture more of the deviant pairs used for structural variant code like Inversion Tool. A lower `inversion.min.qv` value (~10) and a BAM file generated with the "SV" output filter can potentially find a larger number of candidate inversions.

Workflow

Figure 71 shows the inversion tool workflow. The inversion tool begins with a long mate-pair BAM file and proceeds through scoring, pairing, ranking, rescoring, and GFF output.

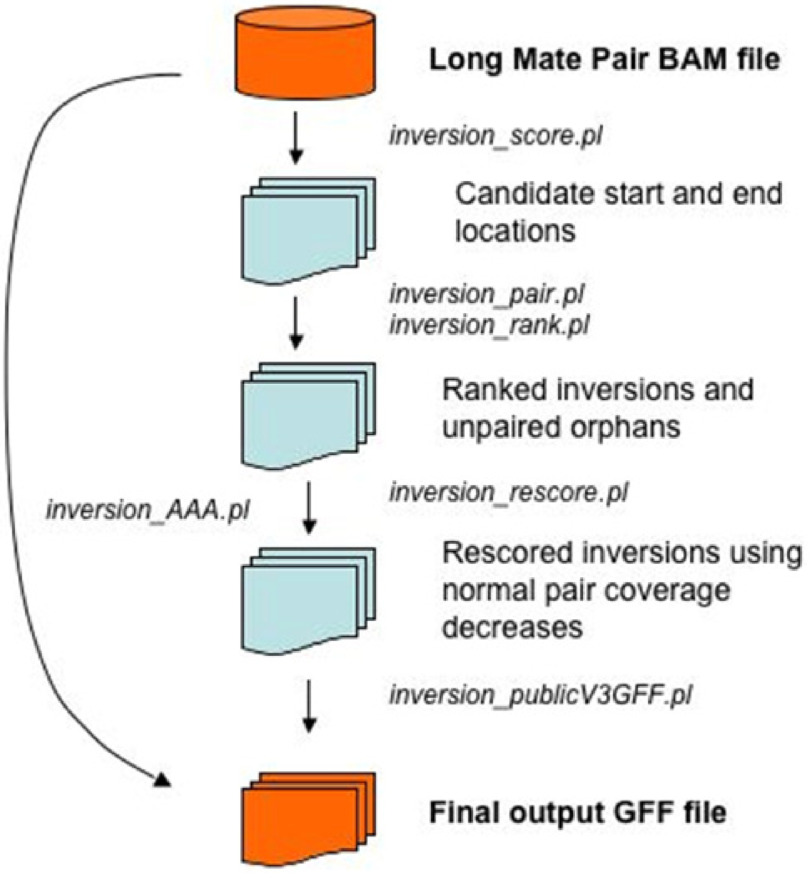


Figure 71 Inversion tool workflow

Scoring

Candidate breakpoints are scored by first determining the tag of a pair that is properly oriented. The inverted mate of the pair is then used to select reference locations that are in the vicinity of the inversion breakpoint. All reference locations that are between the inverted mate and the maximum possible insert size accumulate a positive score. This results in a list of high scoring locations that represent candidate breakpoints.

Pairing

The high scoring candidate breakpoints are then recombined into inversion candidates by matching start and end locations that are defined by an analysis of the high scoring locations. A user definable window size (`breakpoint.peak.width`) is used to determine regions with a peak score greater than the threshold (`breakpoint.score.threshold`). These peaks are combined with their reciprocal nearest neighbor to create a set of possible inversions.

In some cases there will be a peak whose nearest neighbor is slightly below the threshold. A rescue analysis will retrieve those that are significant peaks, but below the threshold cutoff. This is controlled by the `pair.breakpoint.rescue` flag.

Ranking

Inversions are ranked based on the harmonic mean of the scores of the start and end breakpoints. Additionally, inversions are separated out by the user specified maximum inversion length (`max.inversion.length`).

Tiny inversions

If the tiny inversions flag is set (`recover.tiny.inversions`), the pairing and ranking components are rerun with a window size that is sufficiently small to detect inversions with a smaller size (controlled by `max.length.tiny.inversions`).

Normal pair coverage

Additional evidence is provided for breakpoints by examining the relative coverage of unique proper pairs. These are selected from the BAM input using the proper pair flag (0x0002) and are used to reduce the score of reference positions covered by proper pairs.

inversion.ini file example

The following section shows a typical example of the inversion.ini file. For a description of the inversion.ini file parameters, see Table 44 on page 226.

IMPORTANT! Before you begin a run, you must verify the settings for each parameter highlighted in **bold** in the *.ini file example shown in the next section.

```
############################
############################
##
##  global parameters
##
import ../globals/global.ini
reference = ${reference.dir}/
DH10B_WithDup_FinalEdit_validated.fasta

############################
############################
##
##   mapping
##

############################
############################
##
##   pairing
##
mates.file.dir = ${output.dir}/pairing

############################
############################
##
##   inversion
##

# mandatory parameters
# --------------------
# Parameter to specify whether to run inversion Pipeline. 1 .
Run, 0 . Don.t run.
inversion.run = 1

#inversion.mates.list.file or inversion.mates.list.info only one
of the parameters should be used

# Path to the mates.list file
#inversion.mates.list.file=inversion.mates.list

# Comma-seperated, colon-demarcated set of input params, in the
format
# input-file:output-label:[min-clone-insert-length]*:[max-
clone-insert-length]*
inversion.mates.list.info=${mates.file.dir}/F3-R3-
Paired.bam:run1:1000:2000

# optional parameters
# -------------------
#if a value is not provided for output, temp and intermediate
dirs default dirs are created in base directory
# Directory to place output files.
inversion.output.dir=${output.dir}/inversion

inversion.temp.dir = ${temp.dir}/inversion

# Directory to place intermediate files.
inversion.intermediate.dir = ${intermediate.dir}/inversion

# Directory to place log files.
inversion.log.dir = ${log.dir}/inversion

# Whether to calculate normal mate pair coverage around
inversion break points. [0 . Don.t calculate, 1 . Calculate].
Default 0.
#calculate.mp.coverage=

# Number of chromosomes. Default 25.
#no.of.chromosomes=

# ABX score. Default 0.
#abx.score=

# Whether to force updating all intermediate files. [0 . Don.t
update, 1 . Update]. Default 0.
#force.update.intermediate.files=

# Whether to down-weight mate pairs with mismatches
exponentially.[0 . No, 1- Yes]. Default 0.
#down.weight.mp.mismatches=

# Maximal mapped length of BXX matepairs.  Default 3000000.
#max.bxx.mp.length=
# Maximal mapped length of inversions.  Default 100000.
#max.inversion.length=

# Whether to score every run individually. [0 . No, 1 . Yes].
Default 0.
#score.run.individually=

# Whether to pair breakpoints with rescue. [0 . No, 1 . Yes].
Default 0.
#pair.breakpoint.rescue=

# Whether to recover small inversions. [0 . No, 1 . Yes].
Default 0.
#recover.tiny.inversions=

# Maximal mapped length of tiny inversions(implying -tiny,
overriding -maxi)
#max.length.tiny.inversions=

# Break point score threshold.  Default 4.
#breakpoint.score.threshold=

# Output score threshold.  Default 0.
#output.score.threshold=

# Break point peak width.  Default 100.
#breakpoint.peak.width=
```

Inversion tool parameters

Table 44 Inversion parameter description

Parameter name	Default value	Description
Mandatory parameters		
inversion.run	1	Specifies whether to run the tool. Enter 0 if you do not want to run the tool.
inversion.mates.list.file	—	The path to the mates.list file.
inversion.mates.list.info	—	A comma-separated, colon-demarcated set of input parameters in the following format: input-file:outputlabel:[min-clone-insertlength]:[max-clone-insertlength]
inversion.output.dir	—	The path to the directory where the results files will be placed.
inversion.log.dir	—	The path to the directory where the log files will be placed.
inversion.intermediate.dir	—	The path to the directory where the intermediate files will be placed.
inversion.temp.dir	—	The path to the directory where the temporary files will be placed.
Optional parameters		
calculate.mp.coverage	0	Specifies whether to calculate normal mate-pair coverage around inversion breakpoints. Enter 1 to calculate normal mate-pair coverage around inversion breakpoints.
no.of.chromosomes	25	The number of chromosomes.
abx.score	0	The ABX score. ABX is ABA/ABB/ABC, which are mates with both tags on the correct strands but in reverse order.
force.update.intermediate.files	0	Specifies whether to force updating of all intermediate files. Enter 1 to force updating of all intermediate files.
down.weight.mp.mismatches	0	Specifies whether to down-weight mate-pairs with mismatches exponentially. Enter 1 to down-weight mate-pairs with mismatches exponentially.
max.bxx.mp.length	3,000,000	The maximal mapped length of BXX mate-pairs.
max.inversion.length	100,000	The maximal mapped length of inversions.
max.anchor.mismatch	Off	Filter out mates with more anchor mismatches on either tag.
min.alignment.length	Off	Filter out mates with shorter alignment length on either tag.
max.alignment.start	Off	Filter out mates with farther alignment start position on either tag.
score.run.individually	0	Specifies whether to score every run individually. Enter 1 to score every run individually.
pair.breakpoint.rescue	0	Specifies whether to pair breakpoints with rescue. Enter 1 to pair breakpoints with rescue.
recover.tiny.inversions	0	Specifies whether to recover small inversions. Enter 1 to recover small inversions.
max.length.tiny.inversions	—	The maximal mapped length of tiny inversions. Implying - tiny; Overriding - maxi.
breakpoint.score.threshold	4	The break point score threshold.

Table 44 Inversion parameter description *(continued)*

Parameter name	Default value	Description
sab.gff.score.threshold	0	The output score threshold.
breakpoint.peak.width	100	The break point peak width.

Input files

The inversion tool takes one or more BAM files as input. Because the inversion tool specifically selects for pairs that are in the incorrect orientation, it is important to know the correct orientation. As a result, the library type information in the LB field must be properly set.

Output files

Table 45 describes the inversion GFF output file format.

Table 45 The inversion output file format

Column title or number	Description	Example
1	Chromosome.	chr10
2	Method.	AB_SOLiD
3	Feature keywords.	inversion
4	Inversion start coordinate.	46443097
5	Inversion end coordinate.	46479578
6	Inversion score.	129.8
7	Not used.	—
8	Not used.	—
9 Attributes, see below:		
left‡	Starting breakpoint range.	left=chr10:46443097-46443161
right	Ending breakpoint range.	right=chr10:46479540-46479578
leftscore	Left breakpoint range score.	leftscore=185
rightscore	Right breakpoint range score.	rightscore=100
count_AAA_further_left	The number of AAA mate-pairs spanning the genomic region to the immediate left of the starting breakpoint range.	count_AAA_further_left=1
count_AAA_left	The number of AAA mate-pairs spanning the starting breakpoint range.	count_AAA_left=1
count_AAA_right	The number of AAA mate-pairs spanning the ending breakpoint range.	count_AAA_right=2

Column title or number	Description	Example
count_AAA_further_right	The number of AAA mate-pairs spanning the genomic region to the immediate right of the ending breakpoint range.	count_AAA_further_right=1
left_min_count_AAA	Sub-range within starting breakpoint range that has the minimal AAA coverage.	left_min_count_AAA=chr10:46443097-46443112
count_AAA_min_left	Minimal AAA coverage in starting breakpoint range.	count_AAA_min_left=0
count_AAA_max_left	Maximal AAA coverage in starting breakpoint range.	count_AAA_max_left=4
right_min_count_AAA	Sub-range within ending breakpoint range that has the minimal AAA coverage.	right_min_count_AAA=chr10:46479576-46479578
count_AAA_min_right	Minimal AAA coverage in ending breakpoint range.	count_AAA_min_right=4
count_AAA_max_right	Maximal AAA coverage in ending breakpoint range.	count_AAA_max_right=11
homozygous	Whether AAA coverage at both breakpoints ranges are lower than 1/5 of their neighboring ranges.	homozygous=YES

‡ This and the remaining entries are allowed values for Attributes in column 9.

Inversion output file formats

This section provides descriptions of the files produced by the Find Inversions run.

Table 46 Inversion *.gff file format description

File Name/Column	Description	Example
1	Chromosome.	chr10
2	Method.	AB_SOLiD
3	Feature keywords.	inversion
4	Inversion start coordinate.	46443097
5	Inversion end coordinate.	46479578
6	Inversion score.	129.8
7	—	.
8	—	.
9 Attributes, see below:		
left	Starting breakpoint range.	chr10:46443097-46443161
right	Ending breakpoint range.	chr10:46479540-46479578
leftscore	Left breakpoint range score.	185
rightscore	Right breakpoint range score.	100
count_AAA_further_left	The number of AAA mate-pairs spanning the genomic region to the immediate left of the starting breakpoint range.	1
count_AAA_left	The number of AAA mate-pairs spanning the starting breakpoint range.	1
count_AAA_right	The number of AAA mate-pairs spanning the ending breakpoint range.	2

Table 46 Inversion *.gff file format description *(continued)*

File Name/Column	Description	Example
count_AAA_further_right	The number of AAA mate-pairs spanning the genomic region to the immediate right of the ending breakpoint range.	1
left_min_count_AAA	Sub-range within starting breakpoint range that has the minimal AAA coverage.	chr10:46443097-46443112
count_AAA_min_left	Minimal AAA coverage in starting breakpoint range.	0
count_AAA_max_left	Maximal AAA coverage in starting breakpoint range.	4
right_min_count_AAA	Sub-range within ending breakpoint range that has the minimal AAA coverage.	chr10:46479576-46479578
count_AAA_min_right	Minimal AAA coverage in ending breakpoint range.	4
count_AAA_max_right	Maximal AAA coverage in ending breakpoint range.	11
homozygous	Whether AAA coverage at both breakpoints ranges are lower than 1/5 of their neighboring ranges.	YES

Table 47 Inversion all.chr, all.chrx, pair.orphan, pair.txt file format description

Column Title	Description	Example
2	GFF name, type of breakpoint.	InvStart
3	GFF type.	exon
4	Range start coordinate.	1297
5	Range end coordinate.	1362
6	Breakpoint range score, number of supporting mate-pairs.	1
7	Strand.	—
8	—	—
9	—	g=.1
10	Chromosome	chr1

Table 48 inversion rank.txt file format description

File Name/Column	Description	Example
1	Chromosome.	chr1
2	Inversion start coordinate.	1632874
3	Inversion end coordinate.	1706252

Table 48 inversion rank.txt file format description *(continued)*

File Name/Column	Description	Example
4	Inversion score.	4.8
5	Inversion length.	73379
6	Starting breakpoint range.	chr1:1632874-1633166
7	Ending breakpoint range.	chr1:1705726-1706252
8	Left breakpoint range score.	6.0
9	Right breakpoint range score.	4.0

Table 49 Inversion AAA *.gff file format description

File Name/Column	Description	Example
1	Mate-pair bead ID.	1_11_215_288
2	GFF name.	LEFT
3	GFF type.	exon
4	Mate-pair start coordinate.	3971
5	Mate-pair end coordinate.	4020
6	Score, number of mismatches.	1
7	Strand.	+
8	—	—
9	—	g=

Table 50 Inversion coords.inversions.s*.* and coords.orphan.* file format description

File Name/Column	Description	Example
1	The genomic coordinate of an inversion.	chr10:46443097-46479578
2	Inversion score.	129.8

Table 51 Inversion AAA/*.txt and AAA.txt file format description

File Name/Column	Description	Example
1	Chromosome.	chr10
2	Inversion start coordinate.	46443097
3	Inversion end coordinate.	46479578
4	Inversion score.	129.8
5	Inversion length.	36482
6	Starting breakpoint range.	chr10:46443097-46443161
7	Ending breakpoint range.	chr10:46479540-46479578
8	Left breakpoint range score.	185
9	Right breakpoint range score.	100

Table 51 Inversion AAA/*.txt and AAA.txt file format description *(continued)*

File Name/Column	Description	Example
10	The number of AAA mate-pairs spanning the genomic region to the immediate left of the starting breakpoint range.	1
11	The number of AAA mate-pairs spanning the starting breakpoint range.	1
12	The number of AAA mate-pairs spanning the ending breakpoint range.	2
13	The number of AAA mate-pairs spanning the genomic region to the immediate right of the ending breakpoint range.	1

Table 52 Inversion rescore.txt file format description

File Name/Column	Description	Example
1-13	Same as AAA.txt	See Table 51 on page 230.
14	Sub-range within starting breakpoint range that has the minimal AAA coverage.	chr10:46443097-46443112
15	Minimal AAA coverage in starting breakpoint range.	0
16	Maximal AAA coverage in starting breakpoint range.	4
17	Sub-range within ending breakpoint range that has the minimal AAA coverage.	chr10:46479576-46479578
18	Minimal AAA coverage in ending breakpoint range.	4
19	Maximal AAA coverage in ending breakpoint range.	11

Find Inversions results file examples

This section provides examples of the files created when you run the Find Inversions tool.

The following is an example of the file `coords.inversions.s2.w100.100000-`:

```
chr10:46442583-46479737          4
chr21:26295366-26297167          2.7
chr6:168834702-168837413         2.1
```

The following is an example of the file `inversions.s2.w100.100000.GFF`:

```
##gff-version 3
##generated by SOLiD inversion tool
chr10 AB_SOLiD    inversion   46442583    46479737    4     .     .
left=chr10:46442583-46443522;right=chr10:46479431-
46479737;leftscore=6.0;rightscore=3.0;count_AAA_further_left=0;count_AAA_left
=0;count_AAA_right=0;count_AAA_further_right=0;left_min_count_AAA=chr10:46442
583-
46443522;count_AAA_min_left=0;count_AAA_max_left=0;right_min_count_AAA=chr10:
46479431-46479737;count_AAA_min_right=0;count_AAA_max_right=0;homozygous=YES
chr21 AB_SOLiD    inversion   26295366    26297167    2.7   .     .
left=chr21:26295366-26296041;right=chr21:26296491-
26297167;leftscore=2.7;rightscore=2.7;count_AAA_further_left=0;count_AAA_left
=0;count_AAA_right=0;count_AAA_further_right=0;left_min_count_AAA=chr21:26295
366-
26296041;count_AAA_min_left=0;count_AAA_max_left=0;right_min_count_AAA=chr21:
26296491-26297167;count_AAA_min_right=0;count_AAA_max_right=0;homozygous=YES
chr6  AB_SOLiD    inversion   168834702   168837413   2.1   .     .
left=chr6:168834702-168835567;right=chr6:168836564-
168837413;leftscore=2.2;rightscore=2.1;count_AAA_further_left=0;count_AAA_lef
t=0;count_AAA_right=0;count_AAA_further_right=0;left_min_count_AAA=chr6:16883
4702-
168835567;count_AAA_min_left=0;count_AAA_max_left=0;right_min_count_AAA=chr6:
168836564-
168837413;count_AAA_min_right=0;count_AAA_max_right=0;homozygous=YES
```

Prepare to run the Find Inversions tool

Select the required input files

Before you can run the Find Inversions tool you must know the following information:

- The path to at least one *.bam file.
- The Library Type of the primary data exported to BioScope™ Software from the instrument.

Complete the prerequisites

1. Complete the applicable prerequisites described in Chapter 3, "Before you Begin" on page 35.
2. Login to the BioScope™ Software cluster. Change to the working directory and update the inversion.ini file with information that applies to the Find Inversions run. See "Inversion parameter description" on page 226.
3. Complete the resequencing mapping/pairing process on the primary data from the instrument.

Run the Find Inversions tool from the command line

Although several different software programs are involved in the run, a single command generates all of the related programs required to complete the run. The *.plan file that is specified in the command syntax controls the order in which BioScope™ Software runs the related programs.

Start the run

1. Connect to the BioScope™ Software cluster and login with a user ID that has write privileges on all of the directories that BioScope™ Software uses when the tool runs.
2. At a command prompt, enter:
   `bioscope.sh -l` *`filename`*`.log` *`filename`*`.plan`

Do not logout of the BioScope™ Software cluster.

Check the run status from the command line

1. Navigate to the log directory that is defined in the inversion.ini file. For example, you might enter:
   ```
   cd /data/results/tertiary/inversion/log
   ```
2. Open `bioscope.`*`yyyymmddhhmmss`*`.log`
3. Scroll to the end of the file.

The run is complete if you see an entry similar to:

```
15 Apr 2010 03:16:32,537  INFO [main] PluginJobManager:130 -
>>>> END of PluginJobManager >>>> date DURATION=4 minutes 33
secs

15 Apr 2010 03:16:32,537  INFO [main] EventTransportFactory:129
- Closing JMS connection and session
```

Run the Find Inversions tool from the web interface

The instructions in this section assume the following system conditions:

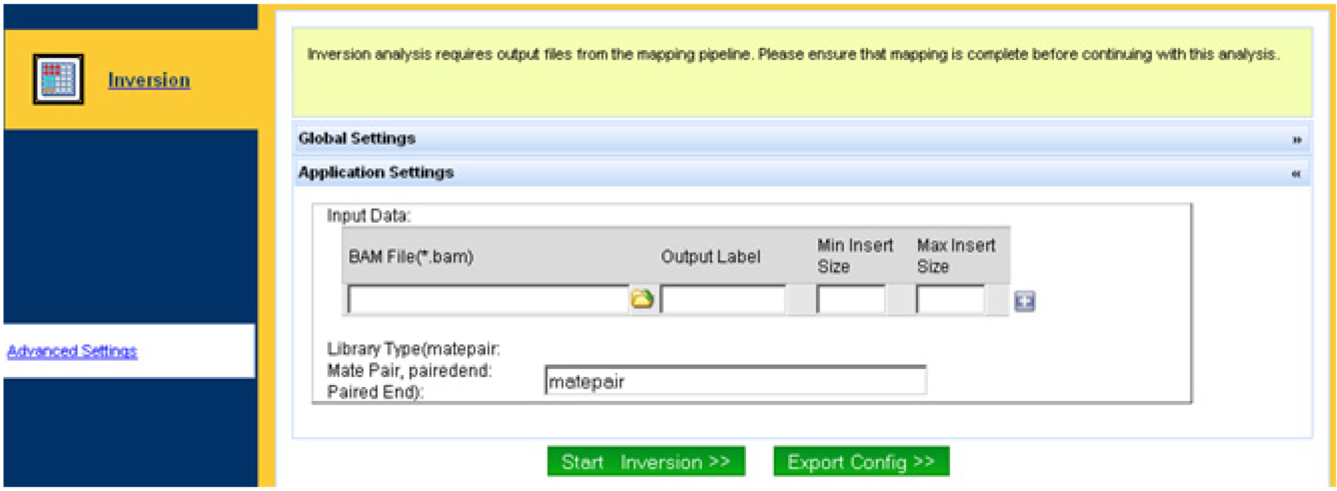
- The Java Messenger, Tomcat, and Apache services are running on the BioScope™ Software cluster.
- You are using Internet Explorer versions 6 or 7 or Mozilla 3.0.1.
- Mapping and pairing is complete.

1. Launch a browser and enter the BioScope™ Software URL:
   http://<hostname>:8080/bioscope

2. Click **Find Inversions**.

The Find Inversions page has two windows and one link (see Figure 72 on page 234):

- Global Settings
- Applications Settings
- Advanced Settings



Inversion analysis requires output files from the mapping pipeline. Please ensure that mapping is complete before continuing with this analysis.

Input Data:

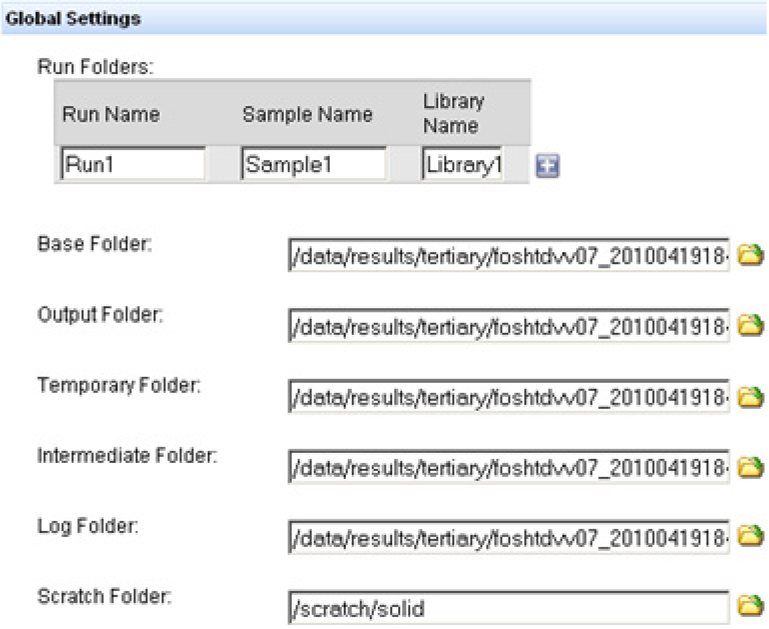
BAM File(*.bam)	Output Label	Min Insert Size	Max Insert Size

Library Type(matepair: Mate Pair, pairedend: Paired End): matepair

Figure 72 Find Inversions page example

Global Settings description

The Global Settings window displays the default values for the folders that BioScope™ Software creates for the files that result from the Find Inversions run (see Figure 73). The window also has fields where you can change the default values for Run Name, Sample Name, and Library Name of the primary data that was exported to the BioScope™ Software cluster from the instrument.



Global Settings

Run Folders:

Run Name	Sample Name	Library Name
Run1	Sample1	Library1

Base Folder:	/data/results/tertiary/foshtdvv07_2010041918
Output Folder:	/data/results/tertiary/foshtdvv07_2010041918
Temporary Folder:	/data/results/tertiary/foshtdvv07_2010041918
Intermediate Folder:	/data/results/tertiary/foshtdvv07_2010041918
Log Folder:	/data/results/tertiary/foshtdvv07_2010041918
Scratch Folder:	/scratch/solid

Figure 73 Find Inversions Global Settings window

Customize the default folder structure (optional)

The folders store the results files generated by each Find Inversions run. BioScope™ Software automatically creates the default folder structure for each Find Inversions run:
`/data/results/tertiary/`*`headnode_yyyymmddhhmmss_x`*

Complete the following steps to change the default directory structure.

1. Click 📂 in the Base Folder field. The File Browser dialog appears.
2. In the **Look in** field, type the custom directory path, for example, `/home/data`.
3. Click **Open**.
4. The folders reflect the updated directory structure.

Note: If you change the default directory structure, the Output, Temporary, Intermediate, and Log folders become subdirectories of the Base Folder.

Update the Run Folder settings (optional)

You can accept the default values in the Run Name, Sample Name and Library Name fields. In this context, "run" refers to the primary data that was exported to BioScope™ Software from the instrument.

To change the default values for the Run Folders:

1. Enter the updated run name in the Run Name field.
2. Enter the updated sample name in the Sample Name field.

3. Enter the updated library name in the Library Name field.

4. Optional: Click ⊞ to add a row for a second run folder.

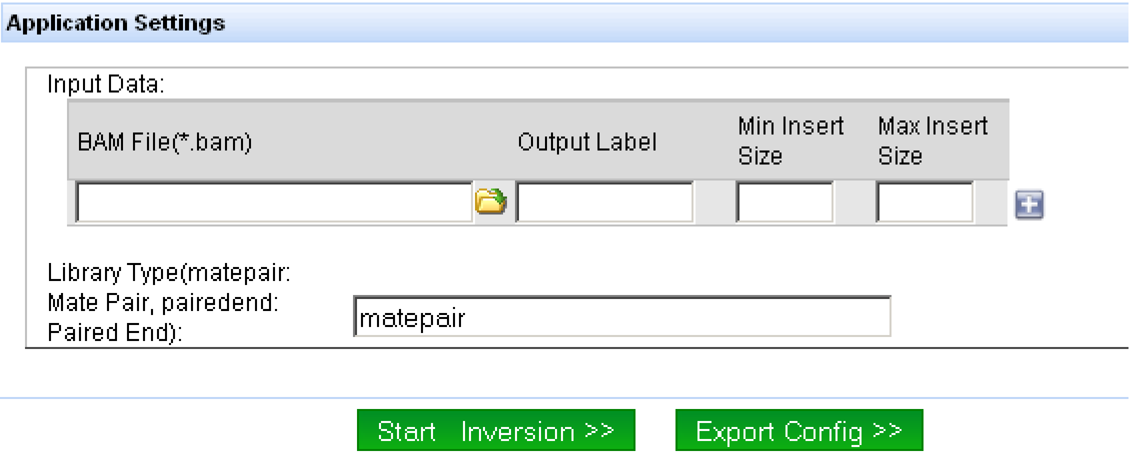
5. Optional: Enter a Run Name, a Sample Name and a Library Name in the new row.

Advanced Settings description

Click **Advanced Settings** to view the current default values defined by BioScope™ Software for the Find Inversions tool. Do *not* change any Advanced Settings unless instructed to by the BioScope™ Software administrator.

Application Settings description

In the Application Settings window (see Figure 74), you must define the absolute path to at least one *.bam file and enter the library type of the data you selected for mapping and pairing. You have the option to enter an Output Label. The text of the output label is used to identify the run in the *.gff files produced by the Find Inversions tool. You can specify the minimum value for clone insertion in the Min Insert Size field, and the maximum value for clone insertion in the Max Insert Size field. You also start the Find Inversions run from the Applications Setting window. The Export Config >> button is only used with the tool that processes barcoded libraries (see Appendix C, "Batch Analysis of Barcoded Library Data" on page 319).



Library Type(matepair: Mate Pair, pairedend: Paired End): matepair

Start the Find Inversions tool run

Figure 74 Find Inversions Application Settings window

1. Click 📂 in the BAM File(*.bam) field. The File Browser window appears.

2. Define the directory path to the *.bam file.

3. Click **Open**.

4. Optional: Click ⊞ to include additional *.bam files.

5. Enter the library type of the primary files that you selected for mapping and pairing. Enter `matepair` if the library type was mate-pair. Enter `pairedend` if the library type was paired-end.

6. Optional: Define an Output Label. The Output Label is displayed in the *.gff file generated by the tool.

7. Optional: Enter the minimum value for clone insertion in the Min Insert Size field.

8. Optional: Enter the maximum value for clone insertion in the Max Insert Size field.

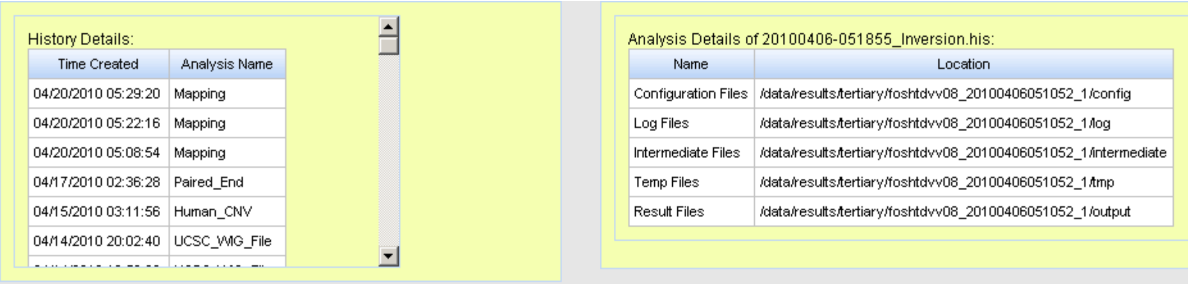
9. Click **Start Inversion >>** to start the run.

10. At the job submission dialog, click **OK** after you have verified the folder locations.

Check the status of the run from the web interface

1. Click **History** . The History window appears and the History Details table is displayed in the left pane. The History Details table shows the Time Created and Analysis Name for all runs performed on the BioScope™ Software cluster.

2. Scroll the History Details table and select an Inversion run, based on the data in the Time Created column (see Figure 75).



History Details:

Time Created	Analysis Name
04/20/2010 05:29:20	Mapping
04/20/2010 05:22:16	Mapping
04/20/2010 05:08:54	Mapping
04/17/2010 02:36:28	Paired_End
04/15/2010 03:11:56	Human_CNV
04/14/2010 20:02:40	UCSC_WIG_File

Analysis Details of 20100406-051855_Inversion.his:

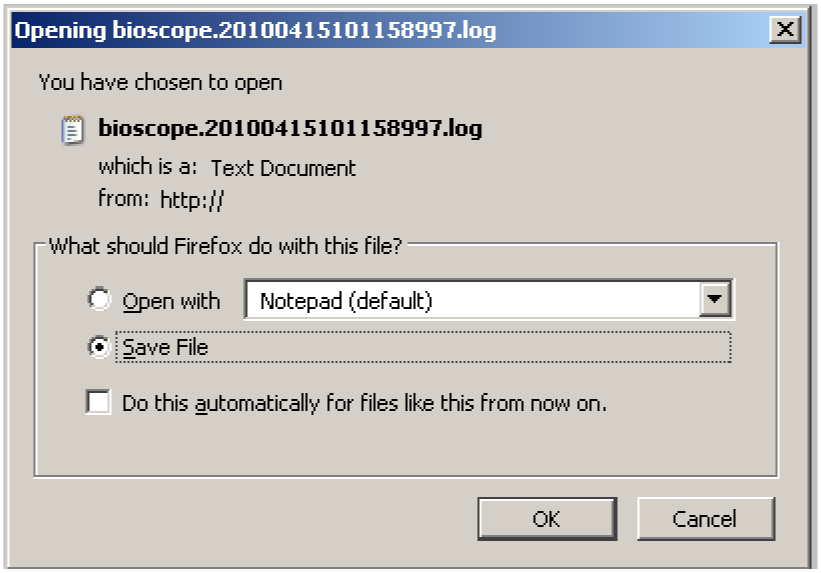
Name	Location
Configuration Files	/data/results/tertiary/foshtdvv08_20100406051052_1/config
Log Files	/data/results/tertiary/foshtdvv08_20100406051052_1/log
Intermediate Files	/data/results/tertiary/foshtdvv08_20100406051052_1/intermediate
Temp Files	/data/results/tertiary/foshtdvv08_20100406051052_1/tmp
Result Files	/data/results/tertiary/foshtdvv08_20100406051052_1/output

Figure 75 History details and analysis details for a Find Inversions tool run

3. Double-click the Log Files row in the Analysis Details table. The File Browser dialog opens. Click **Resend** if your browser displays a message.

4. Select the `bioscope.`*`yyyymmddhhmmss`*`.log` file.

5. Click **Download**.
   - Click **Open with** and click **OK** to view the log file in Notepad or select a different text editor.
   - Click **Save File** to copy the file to your workstation.



Opening bioscope.20100415101158997.log

You have chosen to open

bioscope.20100415101158997.log

which is a: Text Document

from: http://

What should Firefox do with this file?

○ Open with Notepad (default)

● Save File

□ Do this automatically for files like this from now on.

Figure 76 Log file download page example

6. Scroll to the end of the file.

The run is complete if you see an entry similar to:

```
15 Apr 2010 03:16:32,537  INFO [main] PluginJobManager:130 -
>>>> END of PluginJobManager >>>> date DURATION=4 minutes 33
secs

15 Apr 2010 03:16:32,537  INFO [main] EventTransportFactory:129
- Closing JMS connection and session
```

14 Run the Find Large InDels tool

This chapter covers:

Large indel algorithm description

You can use the SOLiD™ 4 system sequencing projects in which small genomic fragments are aligned to a reference. The Large Indel tool works with either paired-end fragments or mate-pair clones to find sets of locus-spanning pairs with significantly deviated insert sizes compared to the average insert size of the entire library.

Large indel analysis overview

Large indel detection is a tertiary tool in BioScope™ Software. Data from the pairing pipeline serve as direct inputs for large indel discovery (see Figure 77 on page 241). Input data formats from previous releases (mates files) can be used with BioScope™ Software, but require an additional reference (*.cmap) file. A *.cmap file is a tab-delimited file containing fields for chromosome name, chromosome index, and a path to *.fasta-formatted references. The *.cmap file is obsolete in BioScope™ Software v1.2 because the *.bam file generated by the mapping pipeline contains the *.cmap field information within the file header.

Analysis is aided if each *.bam file is associated with an optional pairing statistics file. The pairing statistics file is generated as part of the pairing pipeline. The tool tries to auto-detect pairing statistics files for each input pairing directory, and associate accurate pairing parameters to each *.bam file. If the pairing statistics files are absent, the tool estimates the required parameters directly from the *.bam file.

The Large InDel tool processes inputs using an alignment window to hold and analyze sets of locus-spanning pairs. Regions with pairs that have significant insert size deviations are chosen as candidate indel sites, which are processed further to determine zygosity. The output of the analysis is one file in the *.gff format. The *.gff file contains information about the location, size, and significance of each large indel detected by the tool.

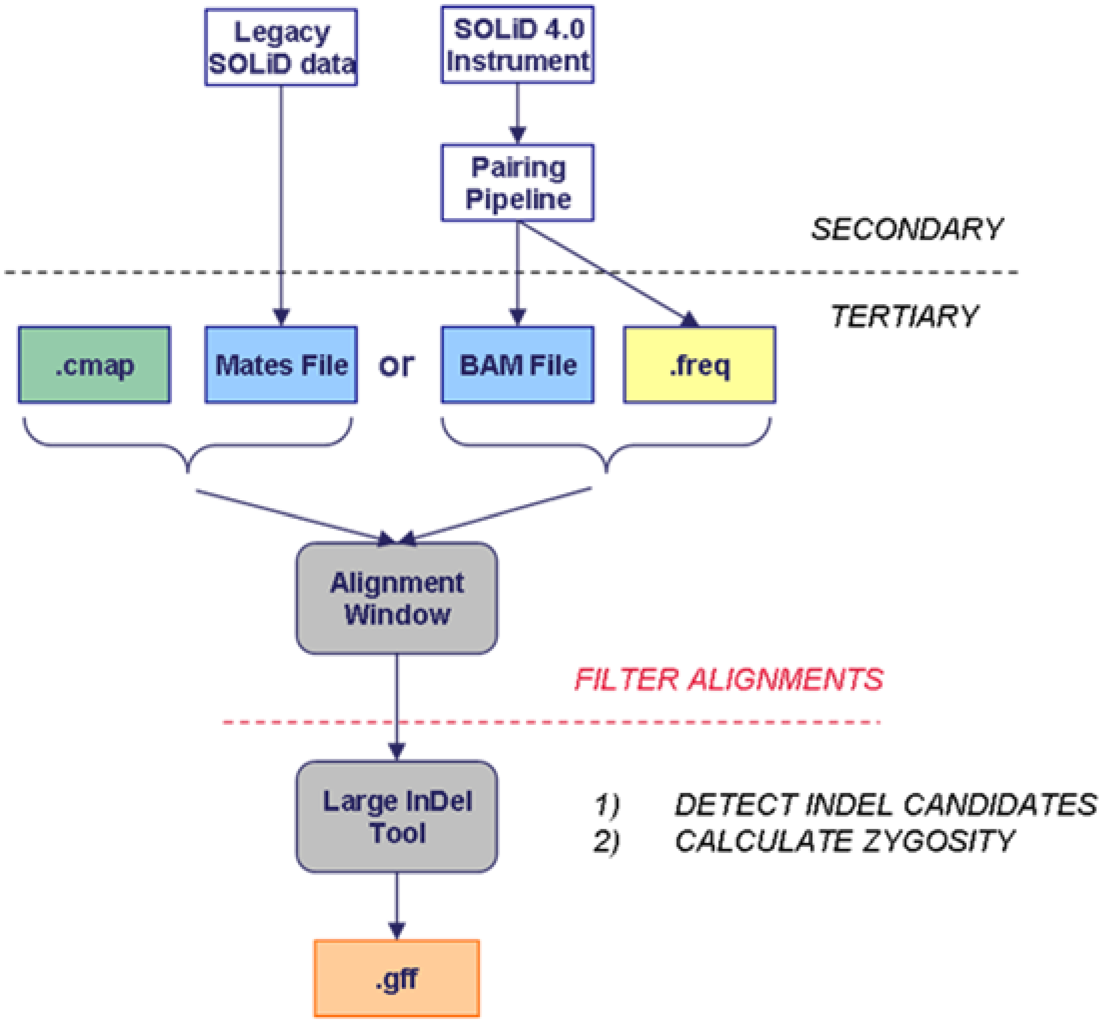


Figure 77 SOLiD™ 4.0 large indel analysis pipeline

This paragraph refers to Figure 77. The large indel tool accepts multiple file types (labeled rectangles) as input. Primary input files (blue) are generated during the secondary analysis pipelines (mapping and pairing). These include legacy mates files and *.bam files. Reference *.cmap files (green) are required for mates files, but not for *.bam files. Pairing statistics files (yellow) generated during the pairing pipeline are ignored for mates file inputs and are optional inputs for BAM files. However, pairing-statistics files provide more accurate pairing statistics and can improve the final results of the Large Indel tool. Final output consists is a *.gff file (orange).

Identify candidate indels

Pairing distances (sometimes called insert sizes) for each pair are assigned during the mapping/pairing pipelines and subsequently used by the large indel tool to determine indel candidacy.

Note: Insert sizes can be non-unique in the case of multiple feasible mapping/pairing combinations. When insert sizes are non-unique, the primary (optimal) pair is chosen for large indel analysis.

Clones that have been mapped and paired to a reference genome can be classified as either concordant or discordant (see Figure 78 on page 242). Concordant pairs are those with insert sizes (sometimes called inter-read distances) that are not significantly deviated from the expected insert size of the library as a whole. Discordant pairs have insert sizes that deviate significantly from the expected value. Discordant pairs containing a putative deletion appear larger when mapped to the reference. Pairs containing a putative insertion appear smaller. Multiple discordant pairs in close proximity provide evidence for a candidate indel within the covered region.

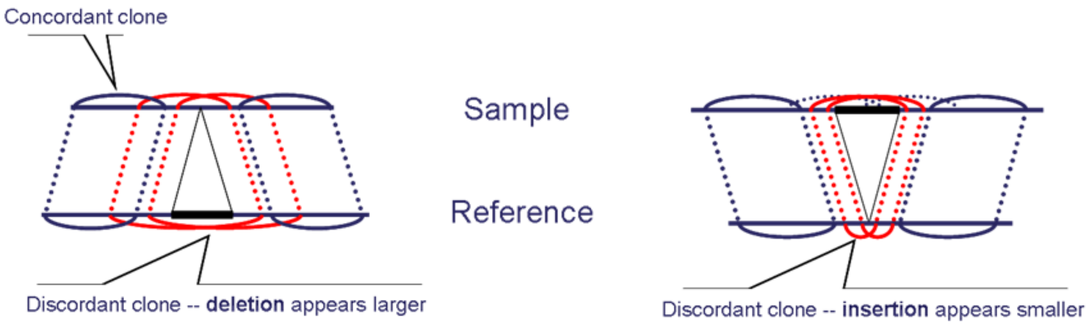

Concordant clone
Sample
Reference
Discordant clone -- **deletion** appears larger
Discordant clone -- **insertion** appears smaller

Figure 78 Discordant clones - used to identify candidate insertions or deletions

In Figure 78:

Pairs spanning a candidate indel (red) appear distorted when mapped to a reference genome. Insert sizes appear larger for deletions (left) and smaller for insertions (right).

The tool moves across individual chromosomes in order of genomic position to generate an alignment window of overlapping locus-spanning pairs (see Figure 79 on page 243). When the window encounters the first read in a pair the corresponding alignment is incorporated into the alignment window. Simultaneously, several moving statistics, including the number of locus-spanning pairs as well as their average insert size and variance, are updated. When the window encounters the second read in a pair, the corresponding alignment is dropped and the moving statistics are updated appropriately. A genomic region (sometimes called a window) is considered to contain a candidate indel when the average insert size of the set of locus-spanning clones is significantly deviated from the average insert size of the library as a whole.

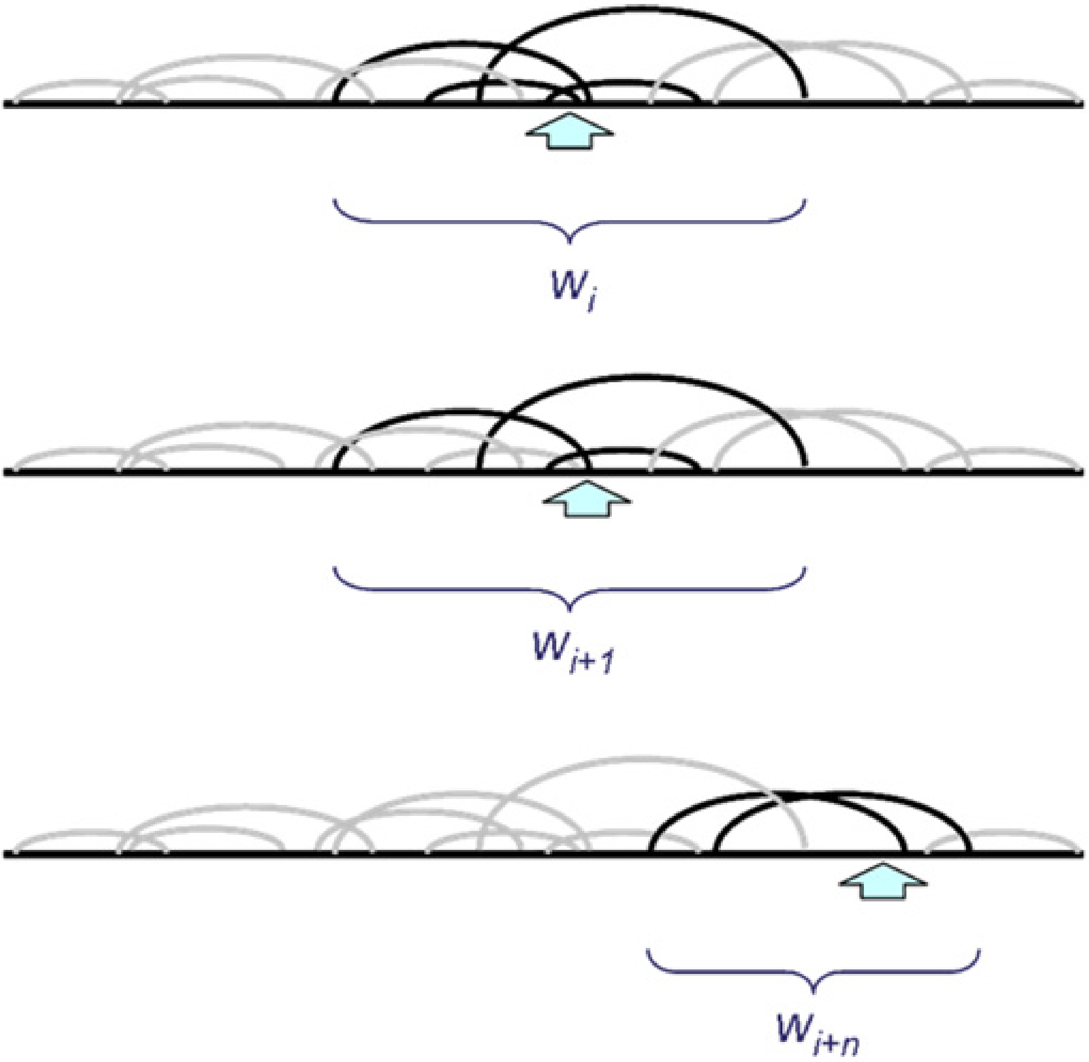


Figure 79 Sets of locus-spanning pairs

The following sections refer to Figure 79.

Top panel

An alignment window, *W*, contains the set of overlapping locus-spanning pairs (black loops) at a particular genomic position *i* (arrow) of the reference genome (black line).

Middle panel

The window advances to the next position by incorporating the next alignment *i+1*. The corresponding alignment is the second in the pair, so it is dropped from the alignment window and the alignment statistics are updated accordingly. When the window encounters the first alignment, the corresponding pair is added to the window.

Lower panel

As this process proceeds along the chromosome, pairs are added or dropped from the window, moving statistics are continuously updated, and regions with significant insert size deviations are detected and analyzed.

Note: Insert size variations are exaggerated for illustrative purposes.

Assigning statistical significance to candidate indels

Regional insert size deviations can be calculated directly from the moving statistics associated with each clone window. The deviations that achieve statistical significance indicate relatively large structural variations compared to the reference genome (see Figure 80). Hypothesis testing determines the significance of deviations, where the null hypothesis asserts an insignificant difference between the local average insert size and the population average $(H_o: x = \mu)$. Candidate deviations are chosen where the probability of falsely rejecting the null hypothesis in favor of the alternative $(H_a: x \mu)$ falls below a user-defined confidence threshold.

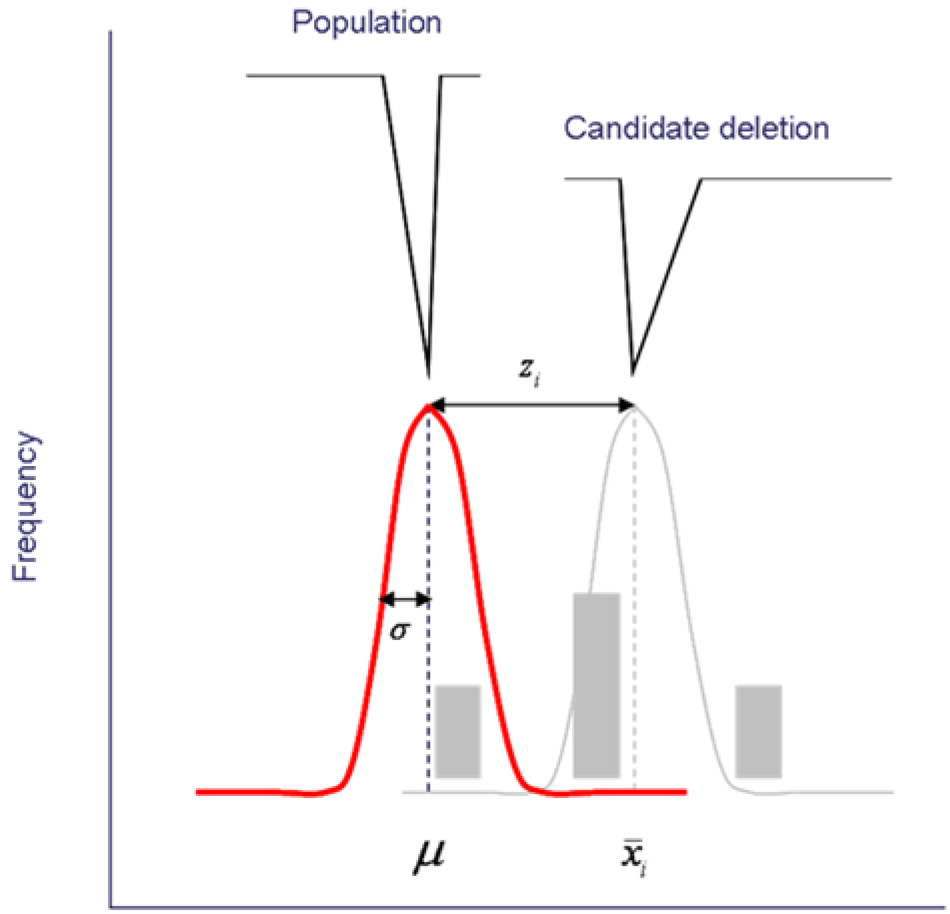


Figure 80 Hypothesis testing example

The following paragraph refers to Figure 80.

The population average insert size μ and standard deviation σ are calculated from the full set of pairs if you use pre-SOLiD 4.0-formatted inputs, or from other sources, such as the .freq file, a *.bam file, or a file directly provided by the user. The alignment window contains a very small subset of pairs sorted by insert size (grey bars). Moving statistics including the number of locus-spanning pairs n_i as well as the sample average insert $\bar{x}_i$ size are calculated from this subset of pairs at each genomic position *i*. The parameters are used to z-normalize insert sizes according to $z_i = \frac{|\bar{x}_i - \mu|}{\sigma}$, which measures the absolute insert size deviation between the sample and the population in units of standard deviation. The normalization step allows multiple libraries with variable insert sizes to be combined into one analysis. A candidate indel is considered significant if $p(z_i|zn_i) < \alpha$ where probability values (p-values) are calculated according to the standard normal distribution and is a user-defined threshold in the large.indel.ini file, where large.indel.p.value default is p=1e-10).

Determine zygosity

After a candidate indel is detected and deemed significant, the alignment window is partitioned to remove erroneous pairs because of mapping/pairing artifacts and to further characterize indel alleles and zygosity (see Figure 81 on page 246). Locus-spanning pairs are partitioned to remove two groups because polyploidy is currently not supported.

Each partition represents a disjoint subset of pairs from the alignment window optimally grouped by insert size so that pairs with similar insert sizes are placed in the same partition. Partitions with only one pair (sometimes called outliers) are removed from consideration. The candidate indel is also removed if the average insert size for pairs in the remaining partition are not significantly deviated from the population average. Removing the candidate indel can occur when mapping/pairing errors are responsible for observed insert-size deviations.

The number of pairs per partition and various summary statistics are calculated for each partition to determine alleles, allele frequencies, and zygosity. Candidate regions with minor allele frequencies less than one-third are removed from consideration.

The Large InDel tool uses a heuristic method to categorize each candidate indel. If both partitions contain pairs with insert sizes that are significantly deviated from the reference but not from each other, the region is designated HOMOZYGOUS. If pairs are significantly deviated from the reference and from each other, the region is designated DOUBLE, which indicates two indel alleles of different types or sizes. DOUBLE regions can be placed into one of several indel/indel categories, for example, insertion/deletion. If one pair is deviated from the reference and the other pair is not, the region is designated HETEROZYGOUS, which indicates the presence of indel and reference alleles.

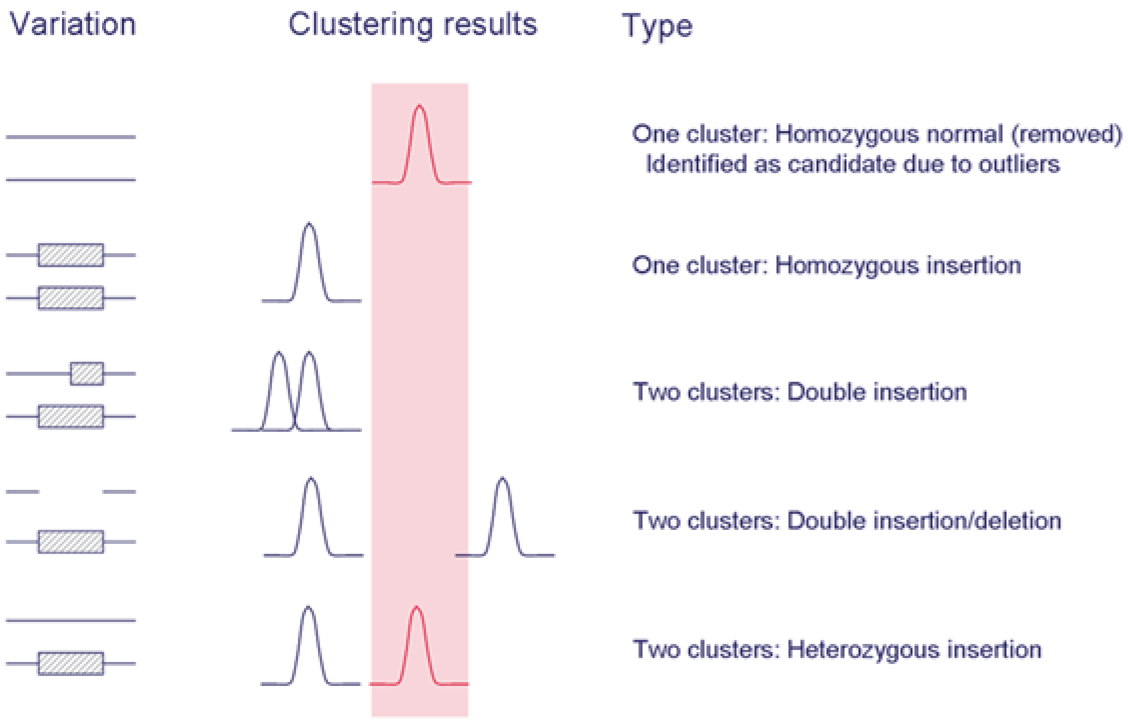

Variation
Clustering results
Type
One cluster: Homozygous normal (removed)
Identified as candidate due to outliers
One cluster: Homozygous insertion
Two clusters: Double insertion
Two clusters: Double insertion/deletion
Two clusters: Heterozygous insertion

Figure 81 Partitioning the alignment window to determine zygosity

The next sections refer to Figure 81.

Left panel

Heuristic methods are used to characterize indel alleles including homozygous reference alleles (lines), which are removed from consideration; insertions (hatched boxes), and deletions (broken lines).

Middle panel

Each partition contains pairs with characteristic insert size distributions. Summary statistics describing the distributions are used to determine indel and reference alleles (black and red bell curves, respectively).

Right panel

Based on the results of the analysis, each candidate indel is assigned an appropriate zygosity category.

Filtering alignments and parameter optimization

A user-defined pairing quality value (PQV) is set in the large.indel.min.pairing.quality parameter in the large.indel.ini file. The PQV setting places constraints on which alignments are incorporated into the alignment window and subsequently used to determine large indel candidate regions. The PQV values are 25 for mate-pair data, and 10 for paired-end data. Adjusting the PQV threshold down (< default) to include lower-quality alignments generally improves sensitivity but increases the number of false positives. The opposite is true if you adjust the PQV threshold up (> default).

Adjusting the p-value and PQV thresholds together might be required to optimize the tool for analysis of data that differ significantly from normal HuRef samples, for example, non-human or cancer genomes. However, the default settings provide a convenient starting point for this process.

Additional optimization might be required for very high coverage samples. For example, alignments from high-density slides mapped to small prokaryotic genomes typically result in clone coverage values above 1000x. In the case of clone coverage values that are above 1000x, use the high-coverage flag (large.indel.high.coverage in Bioscope™ Software), which adjusts the p-value calculation to $p(z_i) < \alpha$, eliminating the conditional coverage parameterization (weighting), and reducing the number of false positives that might otherwise result.

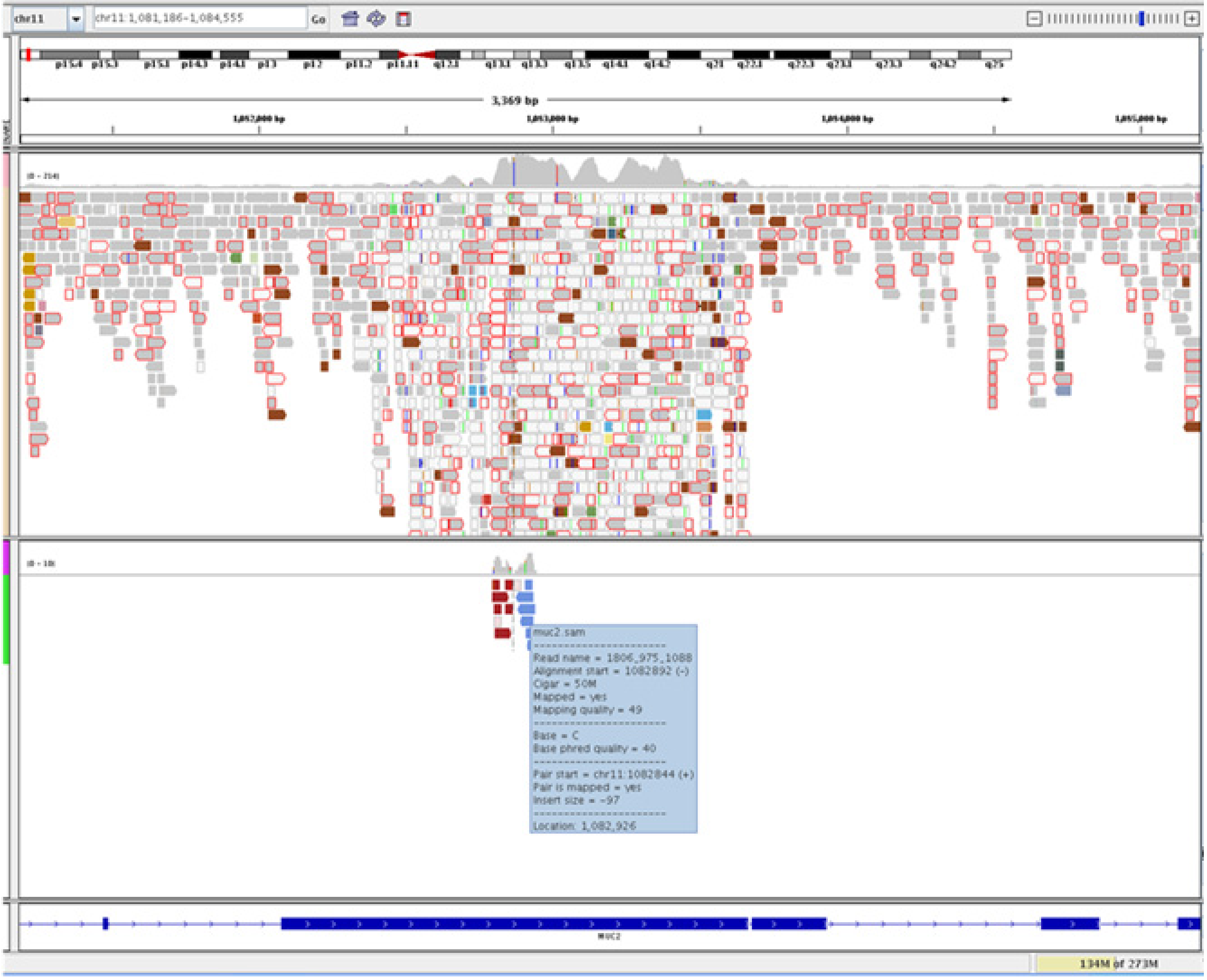
Input files for Large Indel analysis

The only acceptable inputs for Large Indel analysis are *.bam files containing paired-end or mate-pair data. Other alignment types are not compatible with Large Indel analysis. Classic mates files can also be used.

Note: BioScope™ Software v1.2 does not support combining paired-end and mate-pair data into a single analysis.

Interpreting results from the Large Indel tool

Large-indel results are written to a *.gff file, which can be uploaded directly into the UCSC Genome Browser (**cbse.ucsc.edu/research/browser**) or other similar visualization tools. See Table 54 on page 258 for a description of the *.gff file format. See Figure 82 on page 248 and Figure 83 on page 249 for examples of *.gff files generated by the Large Indel tool.

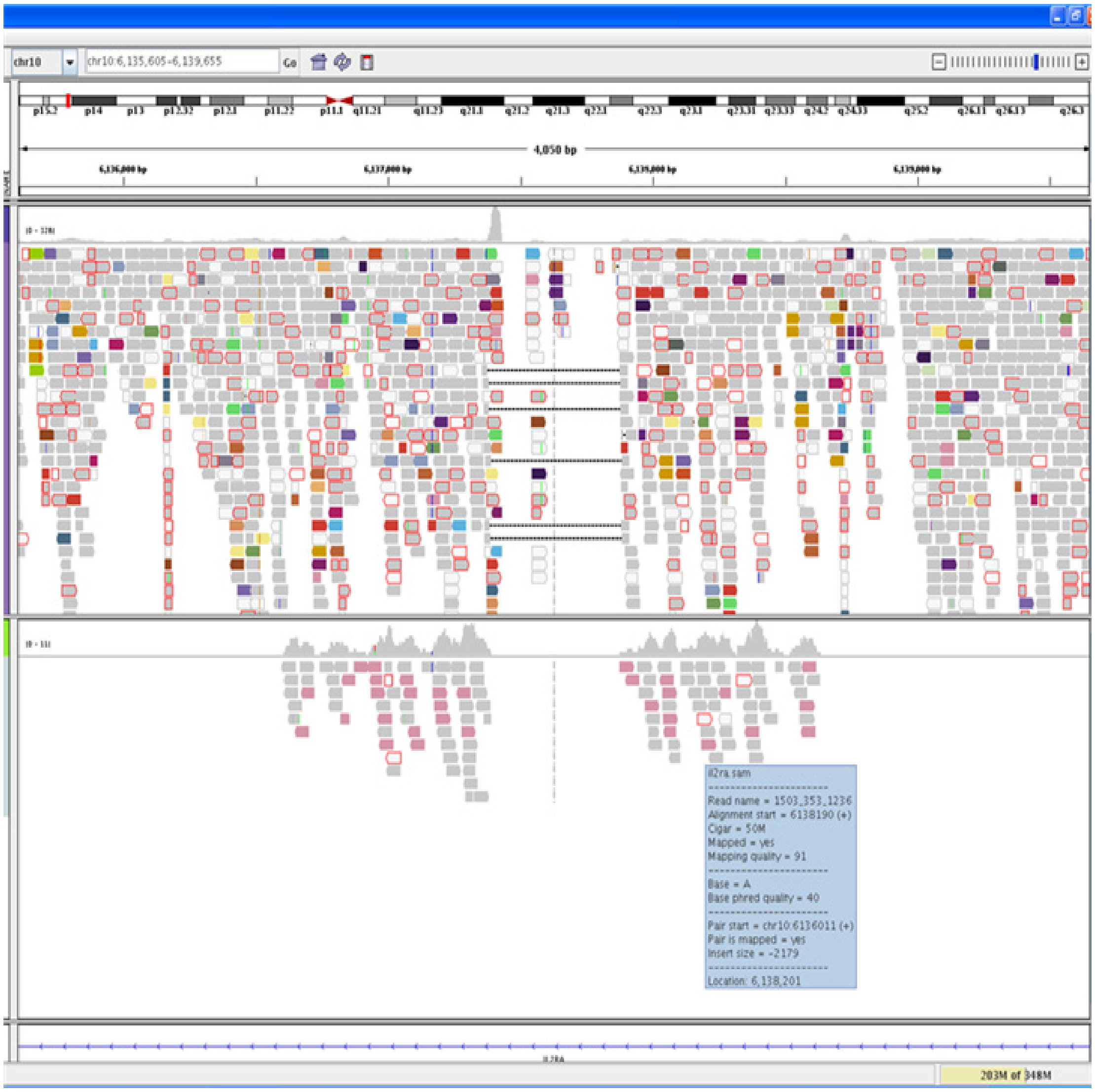


```
muc2.sam
----------------------------
Read name = 1806_975_1088
Alignment start = 1082892 (-)
Cigar = 50M
Mapped = yes
Mapping quality = 49
----------------------------
Base = C
Base phred quality = 40
----------------------------
Pair start = chr11:1082844 (+)
Pair is mapped = yes
Insert size = -97
----------------------------
Location: 1,082,926
```

Figure 82 Identifying a 46bp insertion in *MUC2*

The following paragraphs refer to Figure 82.

Paired-end data was analyzed using the SOLiD 4.0 mapping and pairing pipelines (against a HuRef reference). The resulting BAM file was used for large indel analysis with default settings. The top panel uses the Integrated Genomics Viewer (IGV) to represent all BAM alignments covering the region chr11:1081331-1084700, which contains the partial coding region for the human mucin 2 precursor (MUC2) (bottom panel) as well as an indel previously identified by Levy et al., 2007 (not shown). Further information about IGV, including alignment color-encodings, is available at **broadinstitute.org.** The middle panel represents the subset of alignments used by the Large Indel tool to identify a 46bp homozygous insertion breakpoint at the indicated position (dotted grey line). Forward and reverse strand reads are highlighted red and blue, respectively. The blue rectangle displays various alignment features, including a deviated insert size (97bp) that is significantly smaller than the average insert size of the population (170bp) providing evidence for an insertion at this site. The blue rectangle was generated by mousing over one of the reverse strand reads.



```
il2ra.sam
---------------------------
Read name = 1503_353_1236
Alignment start = 6138190 (+)
Cigar = 50M
Mapped = yes
Mapping quality = 91
---------------------------
Base = A
Base phred quality = 40
---------------------------
Pair start = chr10:6136011 (+)
Pair is mapped = yes
Insert size = -2179
---------------------------
Location: 6,138,201
```

Figure 83 Identifying a 413bp deletion at *IL2RA*

The following paragraphs refer to Figure 83.

The top IGV panel represents long mate-pair alignments (mapped to HuRef) covering the region chr10:6135605-6139655, which contains an intron of human interleukin 2 receptor alpha (bottom panel) as well as several indels previously identified by Ahn et al., 2009; Bentley et al., 2008; Wang et al., 2008; Wheeler et al., 2008; and Levy et al., 2007 (not shown). Several alignment features are consistent with the presence of a deletion at this site, including six gapped alignments (dotted lines) flanking the putative deletion. The region between the six gapped alignments contains very few reads with high mapping quality, indicating sequence that is present in the reference but lacking in the sample. Surrounding the gapped alignments is the set of deviated pairs identified by the Large Indel tool as supportive evidence for a 413bp homozygous deletion (middle panel). Reads with significantly deviated insert sizes, > 2000bp compared to the population average (1575bp), are highlighted (pink).

large.indel.ini file example

The following section shows a typical example of the large.indel.ini file. For a description of the large.indel.ini file parameters, see Table 53 on page 251.

IMPORTANT! Before you begin a run, you must verify the settings for each parameter highlighted in **bold** in the *.ini file example shown in the next section.

```
# To include some common variables.
import ../globals/global.ini
## ********************************************
## pairing parameters
## ********************************************
large.indel/pairing.dir = ${output.dir}/pairing
## ********************************************
## large indel  parameters
## ********************************************
# mandatory parameters
# --------------------
# Parameter specifies whether to run or not large indel
pipeline. [1: to run, 0:to not run]
large.indel.run=1
# Parameter specifies the full path to pairing directory.
Pairing ranges are calculated automatically.
large.indel.pairing.dir=${mates.file.dir}
# Parameter specifies the full path and name of cmap file. Will
be used only when mates.non-redundant is used as input.
cmap=${base.dir}/cmap/test.cmap
# Parameter specifies the full path to the output directory.
# Default value when not specified is 'largeindel'.
large.indel.output.dir=${output.dir}/largeindel
# Parameter specifies the job scripts output directory.
# Default value when not specified is 'largeindel-jobdir'.
large.indel.job.dir=${intermediate.dir}/job-dir

# optional parameters
# -------------------
# Parameter specifies a regular expression, used to find the
distance file (pairing.dat.freq files) generated by pairing
pipelines.
# Default value when not specified or left empty is
pairing\.dat\.freq
#large.indel.freq.file.pattern=
# Parametrer specifies the minimum non matched length
# Default value when not specified is 30
#large.indel.min.map.length=
# Parameter specifies the Levels of clone coverage in lookup
table.
# Default value when not specified or left empty is 1000
large.indel.max.clone.cov=
# Parameter specifies the minimum number of standard deviations
required for significance.
# Default value when not specified or left empty is 6.
```

```
large.indel.min.stdev=
# Parameter specifies the minimum number of clusters used by the
pipeline.
# Default value when not specified or left empty is 2.
#large.indel.min.num.clust=
#
# Parameter specifies the library type: matepair or pairedend.
# Default value when not specified is 'matepair'.
#library.type=
```

Large indel .ini file parameter description

Table 53 Large indel .ini file parameter description

Parameter name	**Default value**	**Description**
large.indel.pairing.dir	Required	The path to the pairing directory. Note: Multiple pairing directories are separated by commas in the large.indel.ini file.
cmap=	This is a required parameter for mates files. Not applicable to *.bam files.	Path to the *.cmap file, for example: /share/apps//etc/cmap/human/cmap
large.indel.output.dir	largeindel/	The path to the output directory.
large.indel.job.script.dir	Intermediate	The job scripts output directory.
large.indel.min.map.length	30	The minimum alignment length for both mate-pair reads. Mate-pairs that do not meet the criteria are ignored. Note: This parameter does not apply to *.bam file inputs. Use large.indel.min.pairing.quality instead.
large.indel.min.pairing.quality	25	Paired reads below this threshold will be ignored. Note: This parameter is only applicable to *.bam file inputs.
large.indel.max.clone.cov	1000	The tableLoci with clone coverage above this threshold will not be analyzed. Note: You can use this parameter in combination with large.indel.high.coverage to reduce false positives in high density genomes, for example, bacteria.
large.indel.p.value	1e-10	P-value threshold, that is, the raw probability of committing a Type 1 error incorrectly identifying a large indel.
large.indel.min.coverage	3	Loci with clone coverage below this threshold will not be analyzed.

Table 53 Large indel .ini file parameter description *(continued)*

Parameter name	Default value	Description
large.indel.high.coverage	disabled	Eliminates the clone coverage weighting when enabled. This option significantly reduces the number of false positives when analyzing very high coverage genomic data. Very high coverage genomic data is typically greater than 1000x read coverage, and is common for bacterial genomes.
large.indel.bas.file	Required for *.bam file inputs only if pairing.dat.freq file does not exist, and the PI field is missing from the *.bam header. Not applicable to *.mates files.	Pseudo-standard file format for storing *.bam file metadata. For details see **ftp://ftp-trace.ncbi.nih.gov/1000genomes/ftp/pilot_data/README.bas**
large.indel.mates.file	[FR][53]-[FR][53]-Paired	A case-insensitive regular expression used to find input data within directories specified by large.indel.pairing.dir. Note: *.bam files, with associated *.bam extensions, are detected automatically and given precedence.
large.indel.freq.file.pattern	pairing\dat\freq	A case-insensitive regular expression used to associate pairing metadata within directories specified by large.indel.pairing.dir.
library.type	matepair	Specifies the library type. Possible values are: • matepair • pairedend

Prepare to run the Find Large InDels tool

Select the required input files

Before you can run the Find Large InDels tool you must know:

- The absolute path to the *.cmap file.
- The absolute path to the Mate Pair mapping results.
- The Mates File Name Pattern.

Complete the prerequisites

1. Complete the applicable prerequisites described in Chapter 3, “Before you Begin” on page 35.
2. Login to the BioScope™ Software cluster. Change to the working directory and update the large.indel.ini file with information that applies to the Large indel run that you want to initiate. See “large.indel.ini file example” on page 250.
3. Complete the mate-pair mapping process on the primary data from the instrument.

Run the Large InDels tool from the command line

Although several different software programs are involved in the experiment, a single command generates all of the related programs required to complete the experiment. The *.plan file that is specified in the command syntax controls the order in which the BioScope™ Software runs the related programs.

Start the run

1. Connect to the BioScope™ Software cluster and login with a user ID that has write privileges on all of the directories that BioScope™ Software uses when the tool runs.

2. At a command prompt, enter:
   `.sh -l` *`filename`*`.log` *`filename`*`.plan`

Do not log out of the BioScope™ Software cluster.

Check the run status from the command line

1. Navigate to the log directory that is defined in the large.indel.ini file. For example, you might enter:

```
cd /data/results/tertiary/log
```

2. Open `.`*`yyyymmddhhmmss`*`.log`.

3. Scroll to the end of the file.

The run is complete if you see an entry similar to:

```
15 Apr 2010 03:16:32,537  INFO [main] PluginJobManager:130 -
>>>> END of PluginJobManager >>>> date DURATION=4 minutes 33
secs

15 Apr 2010 03:16:32,537  INFO [main] EventTransportFactory:129
- Closing JMS connection and session
```

Run the Find Large InDels tool from the web interface

The instructions in this section assume the following system conditions:

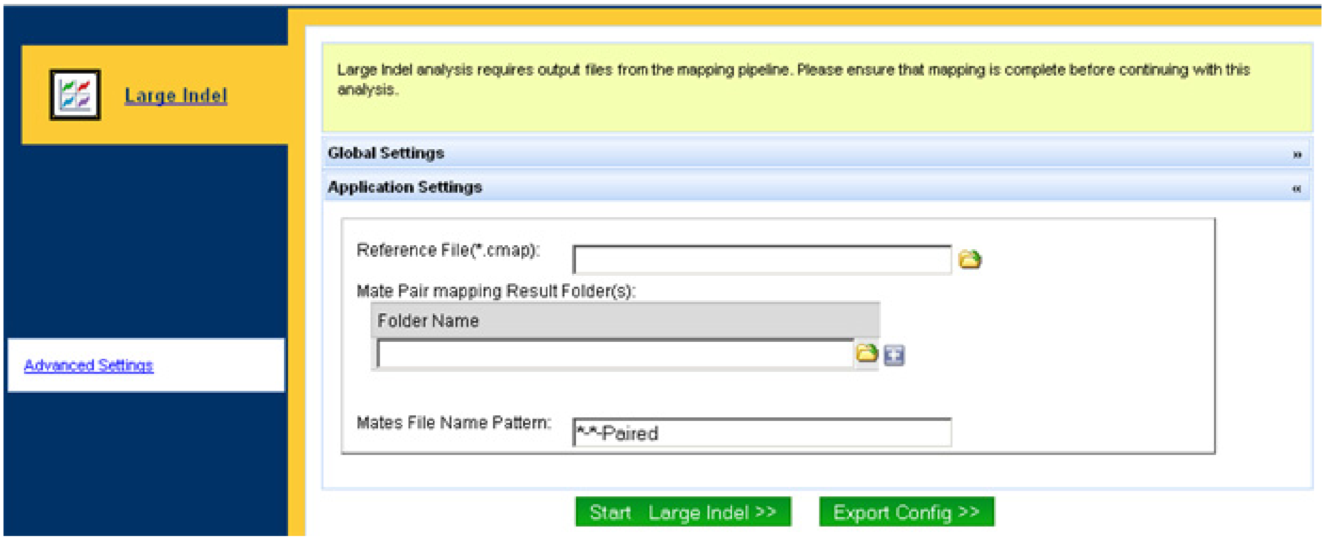
- The Java Messenger, Tomcat, and Apache services are running on the BioScope™ Software cluster.
- You are using Internet Explorer versions 6 or 7 or Mozilla 3.0.1.
- Mate-pair mapping is complete.

1. Launch a browser and enter the BioScope™ Software URL:
   http://<hostname>:8080/

2. Click **Find Large InDels**.

The Find Large InDels page has two windows and one link (see Figure 84).

- Global Settings
- Applications Settings
- Advanced Settings



Large Indel analysis requires output files from the mapping pipeline. Please ensure that mapping is complete before continuing with this analysis.

Reference File(*.cmap):

Mate Pair mapping Result Folder(s):

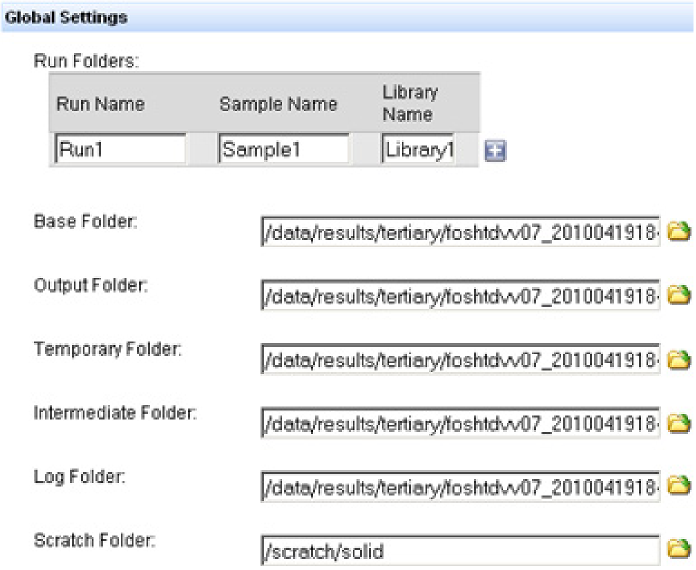
Folder Name

Mates File Name Pattern: *-*-Paired

Figure 84 Find Large Indels page example

Global Settings description

The Global Settings window displays the default values for the folders that BioScope™ Software creates for the files that result from the Find Large Indels run (see Figure 85 on page 254). The window also has fields where you can change default values for the Run Name, Sample Name, and Library Name of the primary data that was exported to BioScope™ Software from the instrument.



Run Folders:

Run Name	Sample Name	Library Name
Run1	Sample1	Library1

Base Folder: /data/results/tertiary/foshtdvv07_2010041918

Output Folder: /data/results/tertiary/foshtdvv07_2010041918

Temporary Folder: /data/results/tertiary/foshtdvv07_2010041918

Intermediate Folder: /data/results/tertiary/foshtdvv07_2010041918

Log Folder: /data/results/tertiary/foshtdvv07_2010041918

Scratch Folder: /scratch/solid

Figure 85 Find Large InDels Global Settings section example

Customize the default folder structure (optional)

The folders store the results files generated by each Find Large InDels run. BioScope™ Software automatically creates the default folder structure for each Find Large InDels run:
`/data/results/tertiary/`*`headnode_yyyymmddhhmmss_x`*

Complete the following steps to change the default directory structure.

1. Click 📂 in the Base Folder field. The File Browser dialog appears.
2. In the **Look in** field, type the custom directory path, for example, `/home/data`
3. Click **Open**.
4. The folders reflect the updated directory structure.

Note: If you change the default directory structure, the Output, Temporary, Intermediate, and Log folders become subdirectories of the Base Folder.

Update the Run Folder settings (optional)

You can accept the default values in the Run Name, Sample Name and Library Name fields. In this context, "run" refers to the primary data that was exported to BioScope™ Software from the instrument.

To change the default values for the Run Folders:

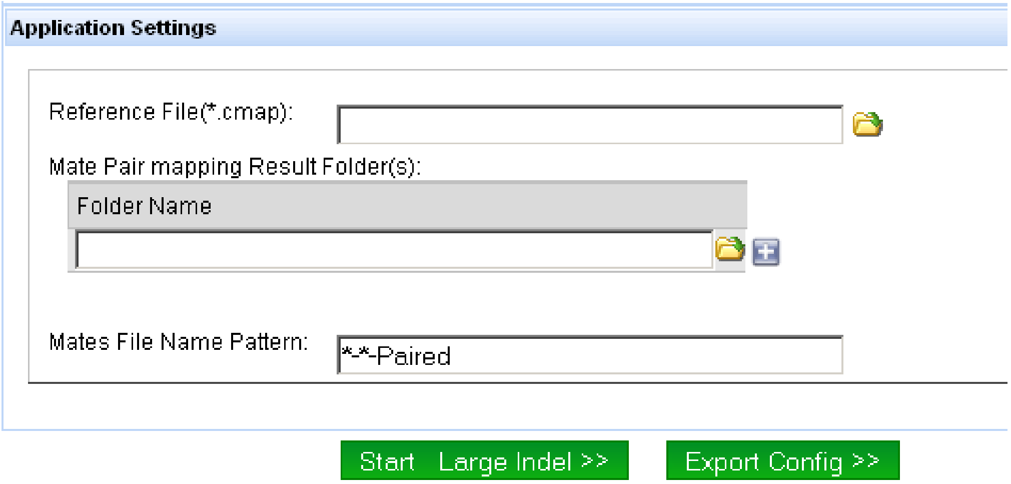
1. Enter the updated run name in the Run Name field.
2. Enter the updated sample name in the Sample Name field.
3. Enter the updated library name in the Library Name field.
4. Optional: Click ⊞ to add a row for a second run folder.
5. Optional: Enter a Run Name, a Sample Name and a Library Name in the new row.

Advanced Settings description

Click **Advanced Settings** to view the current default values defined by the BioScope™ Software for the Large InDels tool. Do *not* change any Advanced Settings unless instructed to by the BioScope™ Software administrator.

Application Settings description

In the Application Settings window (see Figure 86), you must define the absolute path to the *.cmap file. You must also define the absolute path to at least one Mate Pair mapping Result Folder. You must update the parameters each time that you run the Find Large InDels tool. You also start the Find Large InDels run from the Applications Settings window. The Export Config >> button is only used with the tool that processes barcoded libraries (see Appendix C, "Batch Analysis of Barcoded Library Data" on page 319).



Mates File Name Pattern: *-*-Paired

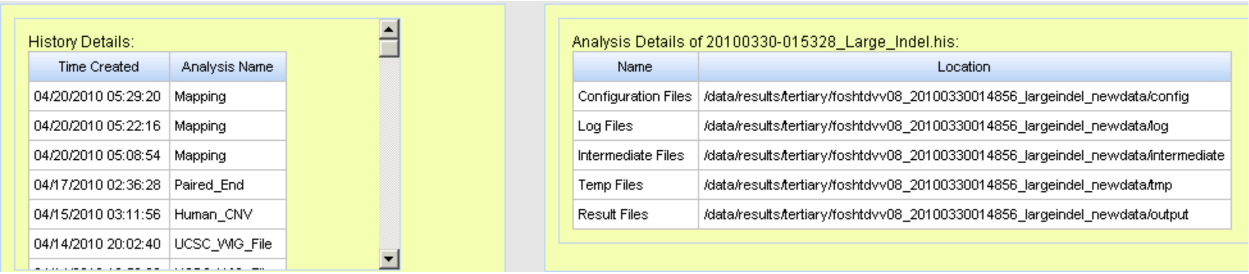
Figure 86 Find Large InDels Application Settings window

Start the Large InDels tool run

1. Click 📂 in the ReferenceFile(*.cmap) field. The File Browser window appears.
2. Define the absolute path to the *.cmap file.
3. Click **Open**.
4. Click 📂 in the FolderName field. The File Browser window appears.
5. Define the absolute path to the folder that contains the mate-pair mapping results.
6. Click **Open**.
7. Optional: Click ⊞ to add a row where you define the absolute path to a second folder that contains mate-pair mapping results.
8. Enter the Mates File Name Pattern.
9. Click Start Large Indel >> to start the run.
10. At the job submission dialog, click **OK** after you have verified the folder locations.

Check the status of the run from the web interface

1. Click **History** . The History window appears and the History Details table is displayed in the left pane. The History Details table shows the Time Created and Analysis Name for all runs performed on the BioScope™ Software cluster.
2. Scroll the History Details table and select the Large_Indel run, based on the data in the Time Created column (see Figure 87).



History Details:

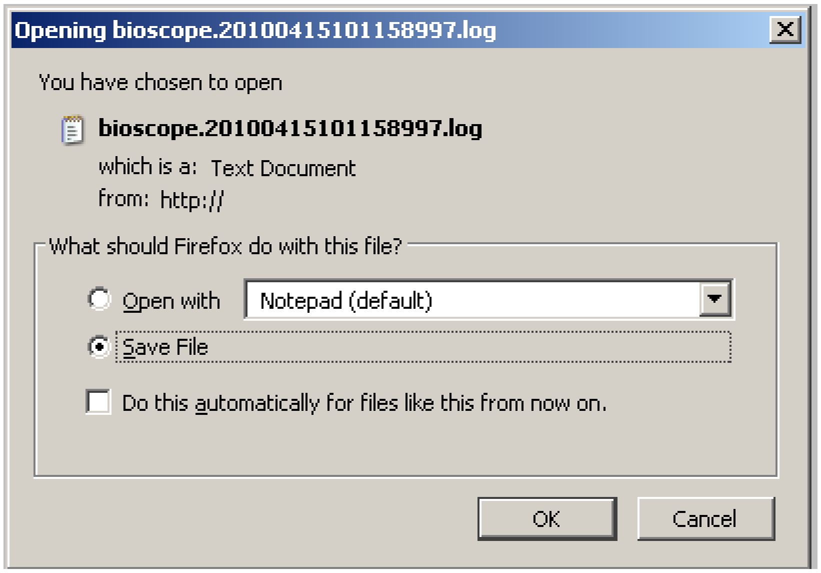
Time Created	Analysis Name
04/20/2010 05:29:20	Mapping
04/20/2010 05:22:16	Mapping
04/20/2010 05:08:54	Mapping
04/17/2010 02:36:28	Paired_End
04/15/2010 03:11:56	Human_CNV
04/14/2010 20:02:40	UCSC_WIG_File

Analysis Details of 20100330-015328_Large_Indel.his:

Name	Location
Configuration Files	/data/results/tertiary/foshtdvv08_20100330014856_largeindel_newdata/config
Log Files	/data/results/tertiary/foshtdvv08_20100330014856_largeindel_newdata/log
Intermediate Files	/data/results/tertiary/foshtdvv08_20100330014856_largeindel_newdata/intermediate
Temp Files	/data/results/tertiary/foshtdvv08_20100330014856_largeindel_newdata/tmp
Result Files	/data/results/tertiary/foshtdvv08_20100330014856_largeindel_newdata/output

Figure 87 History details and analysis details for a Find Large InDels tool run

3. Click **Download**.
   - Click **Open with** and click **OK** to view the log file in Notepad or select a different text editor.
   - Click **Save File** to copy the file to your workstation.



Opening bioscope.20100415101158997.log

You have chosen to open

bioscope.20100415101158997.log

which is a: Text Document

from: http://

What should Firefox do with this file?

○ Open with Notepad (default)

● Save File

□ Do this automatically for files like this from now on.

Figure 88 Log file download page example

4. Scroll to the end of the file.

The run is complete if you see an entry similar to:

```
15 Apr 2010 03:16:32,537  INFO [main] PluginJobManager:130 -
>>>> END of PluginJobManager >>>> date DURATION=4 minutes 33
secs

15 Apr 2010 03:16:32,537  INFO [main] EventTransportFactory:129
- Closing JMS connection and session
```

Large indel output file formats

This section provides descriptions of the large.indel.gff file created by the Find Large InDels run (see Table 54).

Table 54 large-indels.gff file format description

Column Title[1]	Description	Example
Sequence ID	Chromosome name	chr11
Source	Tool name	AB SOLID Large Indel Tool
Type	Indel type	insertion, deletion, and so forth
Start Position	Estimated 5' breakpoint	1081331
End Position	Estimated 3' breakpoint	1084700
Score	Significance of the candidate Indel (p-value)	1e-10
Strand	—	—
Phase	—	—
Attributes[2]		
dev	Indel size, measured in base pair	46
avgDev[3]	Average deviation from the population average	-1.7198
Zygosity[4]	Results from pair partitioning	Homozygous
nRef[5]	Number of reference alleles	0
nDev	Number of deviated alleles	5
refDev[5]	Average deviation of the reference-allele pairs	0
devDev[3]	Average deviation of the deviated-allele pairs	-3.567
refVar[5]	Variance of the reference-allele pairs	0
devVar	Variance of the deviated-allele pairs	0.8972
beadIds[6]	Bead IDs providing support for the candidate indel	1806_975_1088,...

[1] **genome.ucsc.edu/goldenPath/help/customTrack.html#GFF**

[2] Semicolon-separated list of Large Indel tool-specific field names followed by an equal '=' sign, for example dev=46

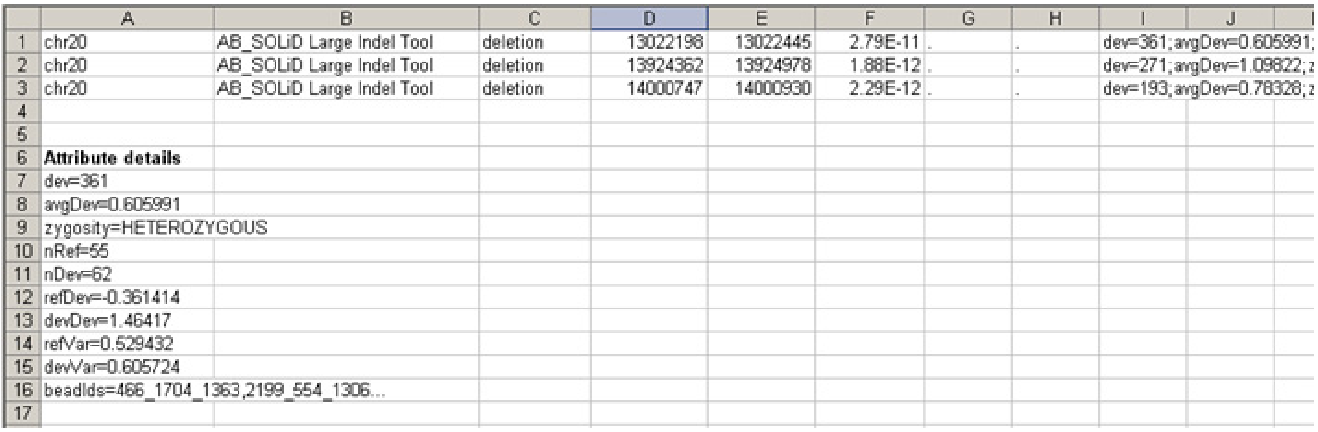
[3] Insertions have negative values

[4] Either homozygous, heterozygous, or double

[5] Value is always zero for homozygous indels

[6] Comma-separated list

Large indel output file example



	A	B	C	D	E	F	G	H	I	J
1	chr20	AB_SOLiD Large Indel Tool	deletion	13022198	13022445	2.79E-11	.	.	dev=361;avgDev=0.605991;	
2	chr20	AB_SOLiD Large Indel Tool	deletion	13924362	13924978	1.88E-12	.	.	dev=271;avgDev=1.09822;z	
3	chr20	AB_SOLiD Large Indel Tool	deletion	14000747	14000930	2.29E-12	.	.	dev=193;avgDev=0.78328;z	
4										
5										
6	**Attribute details**									
7	dev=361									
8	avgDev=0.605991									
9	zygosity=HETEROZYGOUS									
10	nRef=55									
11	nDev=62									
12	refDev=-0.361414									
13	devDev=1.46417									
14	refVar=0.529432									
15	devVar=0.605724									
16	beadIds=466_1704_1363,2199_554_1306...									
17										

Figure 89 large-indels.gff file example

FAQs – Large indels

1

Why are there so many more deletions than insertions?

Insertion detection is limited by the average insert size of the library, typically around 1500 base-pair for mate-pairs. The average insert size of paired-end libraries is much smaller (around 150 base-pair) and insertions are difficult to detect.

2

What is the resolution of Large Indel detection?

This depends on clone coverage, insert size variability, statistical threshold, and library type. For example, mate-pair data detects large insertions and deletions ranging from 30 base-pair to 1.2 kB and 86 bp to 100 kB, respectively, in human hg18 (McKernan et al, 2009). For paired-end data, deletion sizes range from 100 base-pair to 2 kB and insertions are essentially undetectable.

3

Why are there so many large indels around 300 base-pair and deletions around 6 kB?

Alu elements (SINEs) and LINEs, respectively (McKernan et al, 2009).

4

How many large indels can I expect?

The quantity of large indels to expect depends on:

- Genome size
- Clone (read) coverage
- Average insert size
- Significance threshold
- Pairing quality value threshold
- Library type

Note: The Human hg18 has 4075 deletions and 1515 insertions.

5

Why do pooled samples take so long to analyze?

Consider these possible explanations:

- More coverage = more significant candidate regions = more clustering.
- Analysis at CNVs can slow down (coverage increases >2x).

6

How long does the tool take to run and how much space is required?

Consider these possible explanations:

Significant algorithmic improvements have cut processing time. Additional parameters and better implementation have reduced lag times associated with too much coverage and zygosity calculation (clustering).

- Non-parallelized human genome runs consisting of approximately 500 million reads typically take two to four hours to run, depending on platform and resource load.
- Running jobs in parallel at one job per chromosome, significantly reduces run time. Intermediate file generation is negligible for *.bam file inputs.
- Mates files require disk storage equivalent to 1.5 times the input file size.

15 Run the Find Small InDels Tool

This chapter covers:

Small indel detection algorithms

Detection of indels variants using a split-read technique is achieved by using BioScope^{TM} Software's small indel caller using BAM files produced from long mate-pairs, mate-pair, fragment, and pair-end library types. The combination of multiple libraries of these types is also possible. For long mate-pair libraries, the small indel pipeline is able to determine sizes up to 500 for deletions and 20 for insertions. For all other libraries, the size range is up to 11 for deletions and up to 3 for insertions. Furthermore, the pipeline allows for detection of more complex variants such as indel-SNP combinations.

The small indels pipeline determines high-quality calls for insertions and deletions in two stages. In the first stage, the pipeline determines gapped alignments on a bead-by-bead basis. For paired tag libraries, this is performed in BioScope^{TM} Software's pairing pipeline, and for fragment, BioScope^{TM} Software's small indel fragment pipeline described in the mapping chapter (Chapter 9, Run the Resequencing Mapping Tool). In the second stage, the indel caller takes these gap alignments, forms pileups, filters the pileups based on certain heuristics, determines zygosity, and annotates the indel sequences. This results in concisely annotated and highly accurate indel calls.

Paired tag approach

Figure 90 illustrates the F3 with the indel. The algorithm also determines indels in the R3/F5 tag.

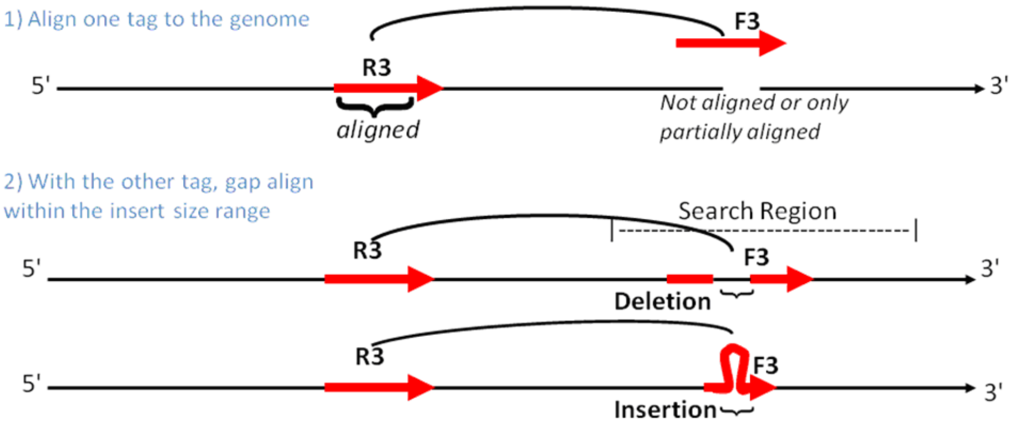


Figure 90 Gap alignment detection using the paired reads from mate-pair and paired-end data

For paired tags, the algorithm surveys only those indels with one-end anchored (OEA) mate-pairs (see Figure 90). It does so by realigning the OEA pairs using an anchor tag (that is one tag which can be aligned the genome by itself) and performing a more aggressive alignment with the other tag in a several kB window (depending on the orientation of the tags and the minimum and maximum insert sizes set in pairing) around the anchored mate. A pair is considered OEA if one read of the tag fails to align or if its aligned length was not higher than a certain length threshold, specified by the pairing parameter `indel.min.non-matched.length`. In addition, anchor tags must also have a minimum anchor length, where its non-match length is also set by this parameter.

Using the unanchored or non-fully extended tag, it starts aligning both ends of the read localized by the other tag's match location. With the region of the genome determined by this match location and insert size distribution, it makes a catalog of end locations by extending the read starting with a minimum value (specified by *i* and *d* parameters in pairing and rescue for insertions and deletions, respectively) until the alignment hits the maximum of number of mismatches allowed. This is specified by these parameters:

- indel.max.mismatches: the number of mismatches in both tags
- pairing.indel.max.mismatch.tag1: mismatches for the F3 tag
- pairing.indel.max.mismatch.tag2: mismatches for the R3/F5 tag

The default value for indel.max.mismatches is 5 and is optimal for 2x50 b.p. reads. However for 2x25 b.p. reads, a setting of 2 is more optimal, and for 2x35 b.p., 3. Additionally, single tag number of mismatches for paired-end libraries is 5 and 2 by default for the F3 and F5 tags, respectively, which is optimal for paired-end libraries.

With this catalog, the process attempts to join the ends of the read to find a single gapped alignment within a certain gap size range. The gap size allowed depends on the size range specified during pairing. Furthermore, it identifies it as a gapped alignment if the above joining could be done with the fewest number of mismatches. Ambiguity of the location of this joining was common, mainly due to the presence of short tandem repeats. However, after indel calling (as described later), this alignment ambiguity is resolved by the consensus of reads.

For determining deletions to size 11 and insertions to size 3, the algorithm disallows for indels within 3 bases from either end of the read. It identifies if it is able to piece together both ends, allowing only for a single gap of up to 4 base pairs inserted (present in read but not in reference), or up to 11 base pairs deleted. To reduce edge effects caused by having a cut-off value in indel finding, the process removes insertions of size 4. These sizes are specified in the pairing.xml and paired-end-pairing.xml files (both found in $BIOSCOPEROOT/etc/plugins/pipelines/), as shown in Table 55.

For long mate-pair (LMP) libraries, different gap size ranges are available (pairing's indel.preset.parameters), and by default all are run in making the pairing BAM file. Table 55 shows the gap size ranges and their representation in the XML.

Table 55 XML representation of gap size ranges

Gap size range	XML representation
Insertions, sized 1 to 3 Deletions, sized 1 to 11	<property name="indel.preset.1" value="D=11,I=4,i=3,d=3" />
Insertions, sized 4 to 14‡	<property name="indel.preset.3" value="D=0,I=15,i=14,Im=4" />
Insertions, sized 15 to 20‡	<property name="indel.preset.4" value="D=0,I=21,i=14,T=3,Im=15" />
Deletions, sized 12 to 200‡	<property name="indel.preset.5" value="D=500,I=0,d=20,Dm=12" />

‡ For long mate-pair libraries only.

For each bead id, multiple matches of the anchor are possible, and each match is considered for determining these gapped alignments. Cases where this results in multiple gapped alignments are removed in the pipeline, so that the final BAM file containing gapped alignments possesses only those alignments having a single good gap alignment.

Single tag approach

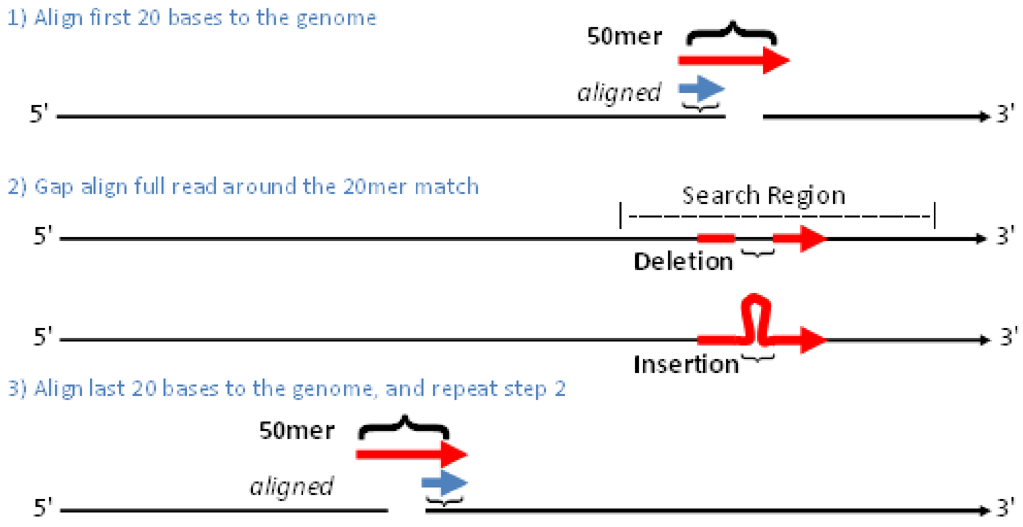

1) Align first 20 bases to the genome
50mer
aligned
5'
3'
2) Gap align full read around the 20mer match
Search Region
5'
3'
Deletion
5'
3'
Insertion
3) Align last 20 bases to the genome, and repeat step 2
50mer
aligned
5'
3'

Figure 91 Gap alignment detection using the single read technique used in fragment libraries and optionally in paired-end libraries

Using only a single tag, gap alignments can also be determined by using small indel fragment pipeline (see "Determining gap alignments" on page 124, in Chapter 9). This is illustrated in Figure 91. As with mate-pair libraries, whole genome searches for gapped alignments would be prohibitively expensive. The strategy taken with fragment libraries is for a particular read, localize it by performing a 20 base pair ungapped alignment allowing for 1 mismatch. The 20mer taken is from both the beginning and end of the read, and only those reads that match to each chromosome less than 100 times are kept for searching gap alignments.

Note: If the indel location is located around the middle of the read, hits from both beginning and end 20 base pair alignments are possible. In these cases, only one is taken into consideration by the downstream indel caller, but both are reported in the `indel-evidence-list.pas` file.

Each 20mer alignment then defines a search region of [A-40, A+80], where A is the position of the alignment. With this region, the same local alignment strategy done with mate-pairs is performed; a catalog of partial begin and end read hits is formed, and an attempt is made to join them with a gap of some size. Similar to the paired tag approach, a read is only considered for this process if the tag fails to align or if its alignment length did not extend past a certain length threshold, specified by the gap aligner's `small.indel.frag.min.non.matched.length`.

For fragment libraries, validated indel sizes are those up to 11 for deletions and 3 for insertions. In a similar manner as in the paired tags, the fragment gap aligner has the `small.indel.frag.indel.parameters` setting, which has the default value of D=11,I=4,d=13,i=10.

Forming and filtering pileups

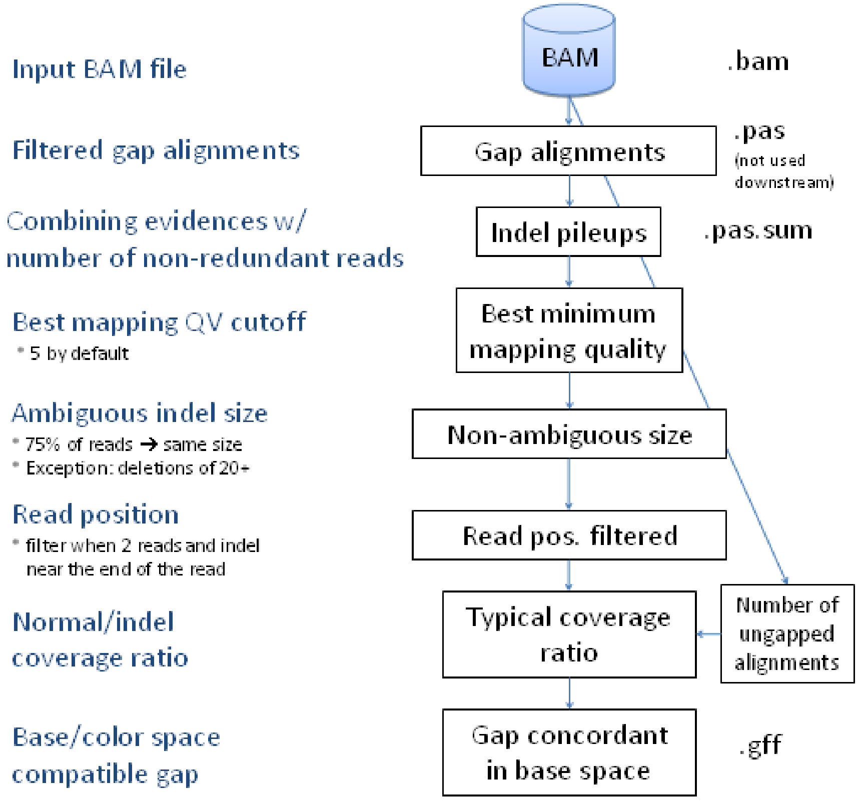

Input BAM file
BAM
.bam
Filtered gap alignments
Gap alignments
.pas
(not used downstream)
Combining evidences w/ number of non-redundant reads
Indel pileups
.pas.sum
Best mapping QV cutoff
* 5 by default
Best minimum mapping quality
Ambiguous indel size
* 75% of reads → same size
* Exception: deletions of 20+
Non-ambiguous size
Read position
* filter when 2 reads and indel near the end of the read
Read pos. filtered
Normal/indel coverage ratio
Typical coverage ratio
Number of ungapped alignments
Base/color space compatible gap
Gap concordant in base space
.gff

Figure 92 Small indel caller heuristics

The gap alignments contained in the BAM files form the basis for calling indels, and goes through a series of processes before being reported in the final GFF output. Extracted from the BAM file are those that have a minimum overall mapping quality (`small.indel.min.mapping.quality`) which is 0 by default. Properties of the gap alignment tag (`small.indel.min.non.matched.length`) and anchor tag (`small.indel.min.map.qv` and `small.indel.min.map.length`) are also assessed and are affected by `small.indel.detail.level`. Only the first 6 (detail level x 2) anchor reads are considered for the default setting of 3, and alternative alignments for the non-matched length filter are considered only for a detail level of 9.

Next the gap alignments are grouped together by genomic location to form pileups of reads. Because of the positional ambiguity of indels, pileups are formed by proximity; specifically, alignments that are within 5 base pairs between consecutive evidences are combined together. For pileups that have 6 or more non-redundant reads, the sixth and additional reads after that will only be grouped together if it is 2 or fewer base pairs from the last gap position. This is the default behavior set by `small.indel.consGroup=1`. A value of 9 will make every gapped alignment into a separate pileup. Finally, the pileup is taken if the number of non-redundant reads is between `small.indel.min.num.evid` and `small.indel.max.num.evid`, inclusively. By default, pileups with two or more reads are taken at this stage. Non-redundant reads are those reads that have a unique F3 and R3/F5 positions for paired tags and F3 for single tag analysis and mixed single and paired tag reads are considered unique from each other. With each pileup extracted from gap alignments, ungapped alignments that span the gap by at least 5 base pairs are counted, and are used in the normal vs. indel coverage ratio. All in all, the system stores the pileup information extracted from the BAM file into an intermediate .pas.sum file for further analysis.

The pileups go through additional filters, as illustrated in Figure 92, each with corresponding optional parameters in Table 56 on page 277. The best mapping QV cutoff is the cutoff for the highest overall mapping quality from the pileup. For paired tags, this is the highest pairing quality value; for fragment, the highest mapping quality, and for mixed, the highest of either. Filtering for the maximum value allows lower quality reads to act as supportive evidence. Ambiguous indel size filtering is for insertions and deletions size 19 and smaller. Also included are pileups that are discordant in size but have at least a certain number of reads with deletions of size 20 or higher. Because no size was determined, these indels are indicated in the GFF with `no_call` for the length. Accurate small indels typically do not have this size ambiguity, whereas larger deletions may.

If only two non-redundant reads are required to make an indel call, false positives can become prevalent at higher coverage levels (for instance, 100x). By using the number of ungapped alignments queried from the BAM file, the software can determine a coverage ratio of ungapped (with 5 base pairs clipped) coverage over a number of non-redundant indel supports. A high ratio is indicative of a false positive and are filtered at values higher than 12 by default and settable by using `small.indel.max.coverage.ratio`. This is marginally helpful in low coverage situations as well.

Finally, the pileup must also be comprised of reads where the majority of reads have gaps that are color space compatible. The parameter `small.indel.colorspace.compatibility.level` sets filtering based on this, where a setting of 0 indicates no color space based filtering and 1 filters out pileups were NO_CALL is most prevalent (1 is the default for BioScope^{TM} Software and recommended for max mapping). NO_CALL for the allele occurs in reads where the gap is not color space compatible.

Color Space Considerations

When an indel occurs in a sequence, and that sequence is measured using color space, the color space sequence not only has a gap of the same size of the indel, but also leaves a signature that can indicate if there is a measurement error within the gap. This is especially important in the case of insertions when you have a small number of evidences and there is a disagreement in the bases of the inserted sequence. With methods that directly measure bases, there would be no indication, based on the inserted sequence alone, on which inserted bases is more trusted.

In color space, this signature can be used to see if the color that spans the gap is compatible with the set of colors that go through the gap. For example, the alignment, AACG/A--G, would be 013/2-- in color space. Here the color 2 spans the gap (measuring both A and G), while 013 goes through the gap (measuring AACG). The color 2 is compatible with the sequence 013 because they both would end with a G in base space. However, an alignment of 213/2-- would not be compatible because, using the same starting base A as the above example, 213 would measure AGTA. Because the rest of the color space sequence beyond this would be aligned, 213/2-- would be indicative of a measurement error within the insertion if 213 is the sample, or if it was the reference, the color 2 that spans the deletion. The alignment's color space compatibility can be calculated for any sequence using color space addition. This signature for color space compatibility is used to more accurately call the inserted sequence of the insertion, important if only a small number of reads were used to call an indel. Also an entire pileup is more likely to be a false positive if most of the reads indicate a gap that is not color space compatible.

Allele calling and short tandem repeat capture

For every gap alignment, the Bioscope^{TM} Software aligner reports the position of the gap and an ambiguity of that placement. The caller takes that information and the color space reads to determine the base space sequence for each of the reads, reporting the reference and all the sequences found. It takes these calls further by taking the reference and the most common allele(s) present and then making a concise representation.

Collecting of allele calls from reads accounting for ambiguous placement

Each read reports a range of positions to represent the ambiguity of the indel placement. Given all the reads in the pileup, the maximum range of positions represented in all of the reads' ambiguity is represented and reported in the gff as the `loose_chrom_pos`. Then for each of the reads, the sequence in this range extended by 1 on the highest position is examined. If they are all the same, the first base and either the last base or the last two bases would be trimmed off. Then all of these reads are collected together and counted. The following tags represent this information:

- `alleles`: the unique sequences found in the reference first, and then the reads
- `allele-counts`: the number of reads with each of these sequences
- `allele-pos`: the position of the first base of the reference sequence in alleles

If there are no reference bases present, the position reported is that of the reference immediately before the insertion. NO_CALL in the alleles tags represents when the gap is not color space compatible (that is, the color or colors spanning the gap would not result in the base space sequence of the reference after the gap).

Here is one insertion example where the indel is called on chromosome 11, position 94446948, corresponding to dbSNP accession rs58864345:

```
Ref   GA---CTTCTTCCC9444694794446956
      21---20220200
Read1  ACTTCTTCTTCCC
Read2  ACTTCTTCTTCCC
Read3  ACTTCTTCTTCCC
Read4  ACTTCGTCTTCCC
```

Since the A in the beginning, and the CC at the end, are the same in all of the reads, the caller would report this in the gff: `allele-pos=94446949;alleles=CTTCTTC/CTTCTTCTTC/CTTCGTCTTCC;allele-counts=REF,3,1`. Although no attempt is made to represent the STR, this non-greedy approach is such that the CTT STR could be captured.

Making concise allele calls

The alleles tag represents a complete picture of all the reads in the pileup. However this is sensitive to outlier reads and is not a compact representation. For these purposes, the caller reports the `allele-call` and `allele-call-pos` tags. This is determined by first taking only the most common indel allele, and the second-most common if it has 75% as many reads as the first. It then greedily trims down the sequence for all the common bases on the right (highest) positions first, then on the left (lowest position). This procedure results in a single “left most” position call which otherwise would result in multiple possible locations.

In the example above, read 4 is removed from consideration. Then CTTCTTC/ CTTCTTCTTC is trimmed down from the right to -/CTT. There’s nothing to trim from the left, so the final call would be `allele-call-pos=94446948;allele-call=/ CTT`.

Examples

Below are several examples that illustrate the allele calling. The positions reported here are from chromosome 1 of the human hg18 reference.

Example 1, a simple insertion:

`ins_len=1;allele-call-pos=55076169;allele-call=/A; allele-pos=55076169;alleles=/A`

An insertion of A is reported here at 55,076,169 because it occurs between position 55,076,169 and 55,076,170.

```
> 4223,chr1:55076169-55076169(),INSERTION,1,;allele-call-pos=55076169;allele-call=/A;allele-
pos=55076169;alleles=/A;allele-counts=REF,5
AATCTATACAGATCATTTCATCTTTTTCTGGCATTGAGT-TATATCTGTACTGGATACCTATGTTTAAGGCTATG 55076131 55076204
0322333112232130021322000022103130122l0-3333221131210233102331100302032331
                               Ref   GT-TA   55076168       55076170
                                     10-3
                               Reads GTATA
                                     1333
  T03331122321300213220000221031301221333333221131210
                   3220000221031301221333333221131210233102331l003022T
                   3220000221031301221333333221133210233102331l003022T
                       000221031301221333333221131210233102331l0030203230T
                       T00221031301221333333221131210233102331l00302032331
```

Example 2, a simple deletion:

`del_len=3;allele-call-pos=91763033;allele-call=AAA/; allele-pos=91763031;alleles=ATAAAGA/ATGA;`

This example shows a deletion of AAA where position 91,763,033 would be the start of the deletion and 91,763,035. The aligner here is representing some similarity around this deletion, which is not an STR.

```
> 7305,chr1:91763033-91763035(),DELETION,-3,;allele-call-pos=91763033;allele-call=AAA/;allele-
pos=91763031;alleles=ATAAAGA/ATGA;allele-counts=REF,2
TGGTGCTGGTGCTCTTAACAATTTTGTAAATAAAGAAGATAATTTCCTTTTCTAGAGGTACAT    91763002       91763064
10113210113222030110300011300330022022330300202000223222013113
                       Ref   ATAAAGA 91763031       91763036
                             330022
                       Reads AT---GA
                             31---2  (Color 1 spans the gap)
  T1321011322203011030001130031---20223303002020002232222
        T132220301103000113003l---202233030020200022322201311З
```

Example 3, an ambiguous insertion:

`ins_len=3;allele-call-pos=2045476;allele-call=/GTA;allele- pos=2045477;alleles=gt/GTAGT;`

In this example there are two possibilities, -/GTA where the insertion is after position 2,045,476 or -/AGT, where the insertion is after 2,045,478. Following the left-most rule, the allele call and allele call position are reported as the lowest chromosome position. The full representation is reported here as gt/GTAGT.

```
> 167,chr1:2045476-2045476(),INSERTION,3,;allele-call-pos=2045476;allele-call=/GTA;allele-
pos=2045477;alleles=gt/GTAGT;allele-counts=REF,2
cacggcggtg---gttagggtcacggctgtag      2045467 2045495
1130330110---10320012113032113 2
  Ref   tg---gtta      2045475 2045479
        10---103
  Reads tGGTAGTTA
        10132103
  T3310110132103200121130321
     T1110132103200121100321132
```

Example 4, insertion of a short tandem repeat:

`ins_len=3;allele-call-pos=5274096;allele-call=/TGT;`
`allele-pos=5274097;alleles=tgtt/TGTTGTT;allele-counts=REF,6`

In this example there is a repeat in the sample, that is not in the reference, so the full reference is TGT/TGTTGT, where TGT in the reference starts at position 5,274,097. Trimming this to the most concise representation, and taking the position immediately before the insertion yields a position of 5,274,096.

```
> 532,chr1:5274096-5274096(),INSERTION,3,;allele-call-pos=5274096;allele-call=/TGT;allele-
pos=5274097;alleles=tgtt/TGTTGTT;allele-counts=REF,6
gttcccatctcctaactggggctaattatcatccctc---tgtttgagtgtttcgaggatgaattgag       5274060 5274124
10200132220230121000323030332132002222---1100122111002322023120301 22
                             Ref   tc---tgtttg       5274095 5274101
                                   22---11001
                             Reads tCTGTTGTTTG
  T20132220230121000323030332132002221101100122111002
0132220230121000323030332132002221101100122111002 1T
     132220230121000323030332132002221101100122111002 32T
T202301210003230303321320022211011001221110023 22023
                T100323030332132002221101100122111002322023120301 22
                T100323030332132002221101100122111002322023120301 22
```

Example 5, deletion of short tandem repeats:

`del_len=4;allele-call-pos=24115702;allele-call=agag/;`
`allele-pos=24115700;alleles=acagagagagaac/aCAGAGAAC;allele-`
`counts=REF,5`

The AG repeat occurs 4 times in the reference and only twice in the sample, or more concisely, AGAGAGAG/AGAG. Because it is a deletion, the left-most position of this repeat is reported here at position 24,115,702.

```
> 2130,chr1:24115702-24115705(),DELETION,-4,;allele-call-pos=24115702;allele-call=agag/;allele-
pos=24115700;alleles=acagagagagaac/aCAGAGAAC;allele-counts=REF,5
agctctgattatgctactgcactccaggctgggtgacagagagagaaccttgacttgaaaaacaaaaCCCCaaaacacagat 24115665       24115746
23222123033132312131122012032100112112222222201020121201200001100010001000111122 3
                            ref    acagagagagaac     24115700       24115711
                                   112222222201
                            reads  aC----AGAGAAC
  T22123033132312131122012032100111 11----2222010201212012
        T3313231213112201203210011211----22220102012120120000 11
                   1122012032100112 11----2222010201212012000011000100 0100T
                          T2210011211----2222010201212012000011000100010000110003
                          T2210011211----222201020121201
```

Example 6, an insertion/SNP combination variant:

```
allele-call-pos=5658680;allele-call=a/CT
```

At position 5,658,680 the reference has an A. The sample however has a CT, so this complex variant is simultaneously a SNP and an insertion.

```
> 593,chr1:5658680-5658680(),INSERTION,1,;allele-call-pos=5658680;allele-call=a/CT;allele-
pos=5658680;alleles=atttt/CTTTTT/CTTTTA/NO_CALL;allele-counts=REF,2,1,1
gtgctgatcagtatttagctgaagactctggaga-tttttgttttgtgactttgtccttttc      5658647 5658707
1132123212133003232120221222102223-000011000111212001120200002
                        ref    aga-tttttg     5658678 5658685
                               223-00001
                        reads  aGCTTTTTTG
  221232121330032321202212221022320000011000111212 03T
          1330032321202212221022320000011000111212001120 2223T
   2011212133003232120221222102232000331100011121 2003T
```

Example 7, a non-adjacent, Insertion/SNP combination variant:

```
allele-call-pos=3686628;allele-call=tct/AGTCC;
allele-pos=3686624;alleles=agagtct/AGAGAGTCC;
```

Starting after position 3686623, there is an AG repeat, followed by TC, and then a T/C SNP. The indel caller combines these into a single allele call, TCT/AGTCC, at position 3686628. Without the SNP, the caller would identify this variant as allele-call-pos=3686623;allele-call=/AG.

```
> 362,chr1:3686623-3686623(),INSERTION,2,;allele-call-pos=3686628;allele-call=tct/AGTCC;allele-
pos=3686624;alleles=agagtct/AGAGAGTCC;allele-counts=REF,5
cgtgtgcttagccgctgctgtgtgatcac--agagtctttacacaagcctcgatggtgcatgtagttttat 3686595 3686663
311113203230332132111112321 11--22212200311110230223231011313113210003 3
                           ac--agagtcttt         3686622 3686631
                           11--22212200
                           aCAGAGAGTCCTT
                           112222212020
  T013203230332132111112321112222212020311110230223 23
             T01321111123211122222120203111102302232310113131132
                   T111232111222221202031111023022323101131311321003 33
    132032103321321111123211122222120203111102302 23233T
                3111111232111222221202031111023022323101131 3113213T
```

Example 8, multiple inserted alleles:

```
ins_len= 3;allele-call-pos=60612130;allele-call=/TAG/
TTG;allele-pos=60612129;
alleles=AT/ATTAG/ATTTG/NO_CALL;allele-counts=REF,4,3,1;
experimental-zygosity=HOMOZYGOUS;experimental-zygosity-
score=0.0038
```

This example has two main inserted alleles, TAG and TTG. The experimental zygosity call is done solely with respect to having the presence of the reference allele since it is calculated by counting the number of reads that span the breakpoint. Because of this it is not HEMIZYGOUS, however, this call is actually has two different alleles that differ only by a SNP, but because it is an insertion with respect to the reference it gets classified as an indel.

```
TGGGGGATAAGGTGTTTATTAAGGATGACAGAAACCTCCTGAT---AGAGACAATATCATTCACCTTATAGATCCATCTCTG 60612088
10000233020111003303020231211220010220212 33---2222110333213021102033322320132222 1
                                 Ref   TGAT---AG     60612127       60612131
                                       1233---2
                                       12---332
                                 Read1 TGATTTGAG
                                       12300122
                                 Read3 TGATTAGAG
                                       12303222
  T10233020111003303020231211220010220212300122222110
  T10233020111003303020231211220010220212300122222110
   T1233020111003303020231211220010220212303222222221103
                3303020231211220010220212303222222211033321302110 23T
                              T30102202123032222222110333
                              T301022021230322222221103332130211020333223201322222
                                     T0123001222222110333213 0211
```

Heterozygous calling

The aforementioned coverage ratio also serves to call an indel hemizygous. This type of zygosity detected is one where one allele is the reference, and the other contains an indel. Because the coverage ratio is of the gapped alignments over the ungapped, this is an indicator of this situation. The software contains a table of coverage ratios, number of times a homozygous (more accurately, non-hemizygous) was observed, and number of times, hemizygous in a file located at `$BIOSCOPEROOT/etc/small-indels/zygosity-calibration.conf`. The table was derived by matching DH10B reads to an indel introduced reference to simulate the non-hemizygous state (indels of the same length on both alleles). For the hemizygous state, reads that occurred over a simulated indel had a 50% chance of being altered to contain that simulated indel. Because indel alignments are generally less sensitive, hemizygous situations occur frequently above coverage ratios values above 1. Different situations were simulated and are available using the parameter `small.indel.zygosity.profile.name`.

Resequencing workflow for small indels

In order to call indels with this pipeline, certain resequencing pipelines are required. The following provides an outline of these workflows for the supported library types. Mate-pair libraries will use the paired tags workflow, while fragment will use the single tag workflow. Paired-end can be analyzed using any of the workflows below. Multiple libraries can also be combined together by producing a BAM file for each of the libraries, and specifying them

Paired tags

The paired tag approach is for a single slide with a library type such as mate-pair and paired-end. An example plan file for the paired tag approach is:

```
= mappingF3.ini
= mappingR3.ini
pairing.ini
= smallIndel.ini
= otherVariantCallers.ini
```

This aligns to the genome with and without gap alignments (mapping/pairing) and then performs indel calling for deletions up to 500 and insertions up to 20 (see Figure 93).

Single tag

The single tag approach is for a single slide of a fragment library type. An example plan file for the single tag approach is:

```
mappingF3.ini
smallIndelFrag.ini
maToBam.ini
= smallIndel.ini
= otherVariantCallers.ini
```

The combined approach accomplishes these tasks:

1. Aligns to the genome without gaps.
2. Aligns to the genome with gaps.
3. Determines mapping stats.
4. Combines the gap and ungapped alignments into a single chromosome-sorted BAM file.
5. Generates a separate BAM file of unmapped reads.

This approach finds deletions up to 11 and insertions up to 3 (see Figure 93).

Combined approach (optional for paired-end)

A combined approach is an alternative method for a single slide of paired-end data. An example plan file for the combined approach is:

```
= mappingF3.ini
= mappingF5.ini
pairing.ini
smallIndelFrag.ini
maToBam.ini
+ smallIndel.ini
```

The combined approach accomplishes these tasks:

1. Aligns both the F3 and F5 tags to the genome.
2. Produces a BAM file for the F3-F5 pair.
3. Produces a BAM file for just the F3 tag.
4. Performs small indel calling on the both BAM files.

The combined approach is specified in the smallIndel.ini file as `small.indel.bam.file=${output.dir}/pairing/F3-F5-P2-Paired.bam,${output.dir}/maToBam/f3.csfasta.ma.bam`. This finds deletions up to 11 and insertions up to 3 (see Figure 94). This method yields greater sensitivity, but may negatively impact the experimental heterozygous calling.

Multiple slides of data

The small indel caller has the ability to use information from multiple slides to gain more power than calling indels on each of the slides separately. The slides can be from any library type. To do this, simply run the secondary analysis on each of the slides of interest, and then run the caller once on all of these inputs simultaneously.

An example parameter input in the small indel ini file for multiple data slides is:

```
small.indel.bam.file=${output.dir}/pairing1/F3-F5-Paired.bam,
${output.dir}/pairing2/F3-F5-P2-Paired.bam,${output.dir}/
maToBam/f3.csfasta.ma.bam,${output.dir}/pairing3/F3-F5-Paired
.bam
```

This parameter setting illustrates how to combine results from multiple BAM files (see Figure 95).

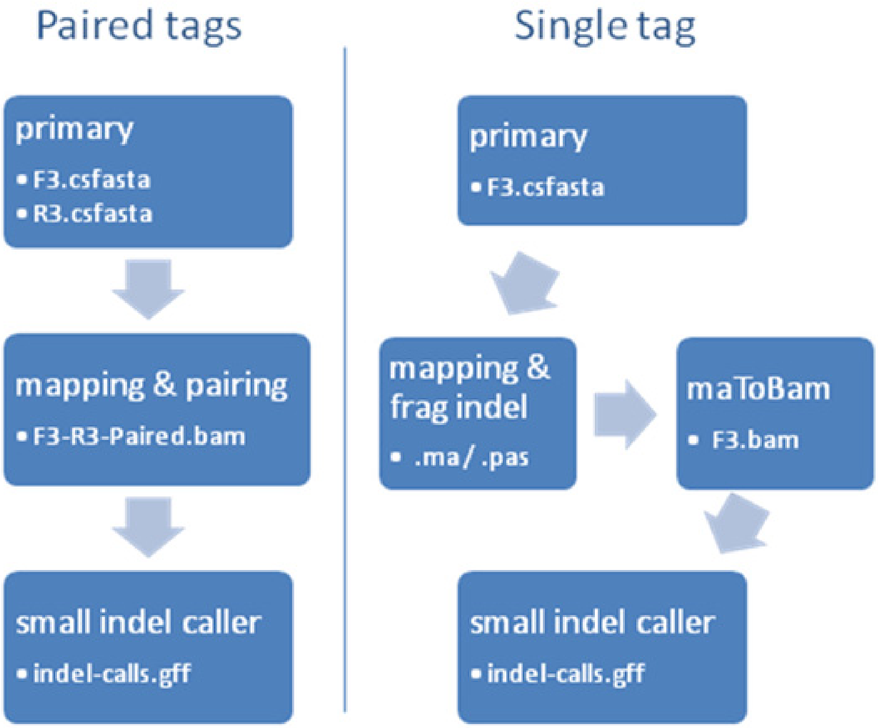


Figure 93 Small indel workflows for paired tags and single tag libraries

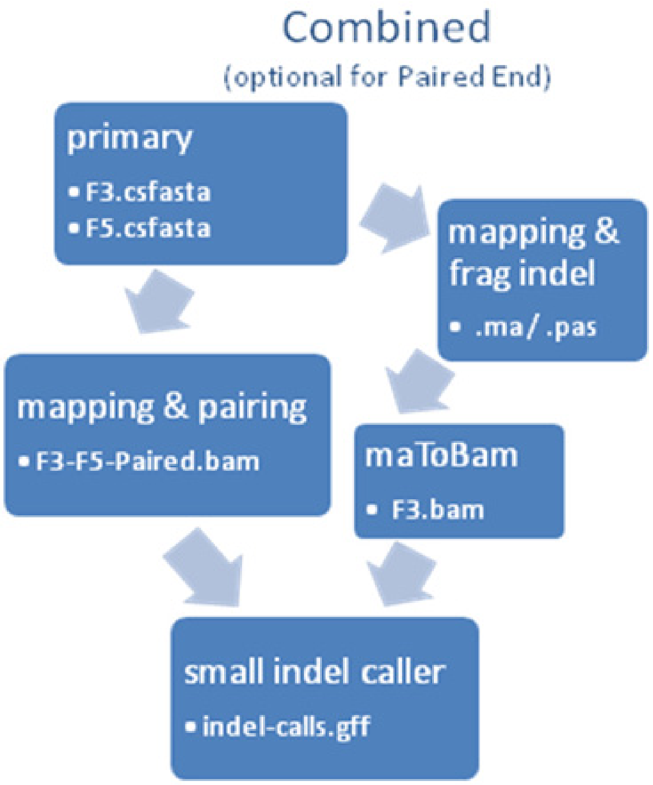


Figure 94 Small indel workflow for the combined tag approach

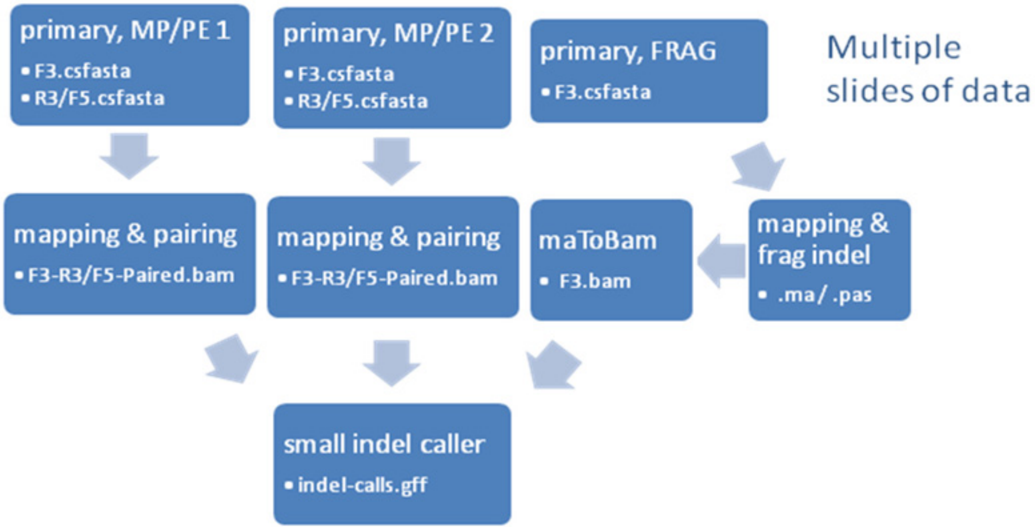


Figure 95 Small indel workflows for combining multiple slides of data together to call a single list of indels

small.indel.ini file example

The following section shows a typical example of a small.indel.ini file. See Table 56 on page 277 for a description of the small.indel.ini parameters.

IMPORTANT! Before you begin a run, you must verify the settings for each parameter highlighted in **bold** in the *.ini file example shown in the next section.

```
# Global parameters (also can be specified in a global.ini)
base.dir=.
output.dir=../../outputs

# Bioscope run command for small indels
small.indel.run=1

## Required parameters
# Input BAM file
# BAM file can come from pairing (i.e. PE, LMP) or from maToBam
(i.e. FRAG)
# For multiple runs, separate with a comma, and no space, i.e.
# ${pairing.file.dir}/F3-R3-Paired.bam,${maToBam.file.dir}/
solid.csfasta.ma.bam
small.indel.bam.file=${output.dir}/pairing/F3-F5-P2-Paired.bam

# Required chromosome information
cmap=/share/apps/Bioscope_dev/BioScope-1.2.rBS120SRN-
46946M_20100423093314/etc/cmap/human.cmap
# As a onetime setup, this file will need to be edited.
# See "CMAP file format description" section for details.

# Optional output directory and prefix filename
small.indel.candidate.dir=${output.dir}/small-indels
small.indel.output.prefix=solidRun_20100426-PE

## Optional parameters
# Memory request in mb, default is 3800.  Request more when
combining very large sets.
memory.request=15000
```

Small indel .ini file parameter description

Table 56 Small indel .ini.file parameters

Parameter name	Default value	Description
small.indel.run=1		Run this pipeline.
Mandatory input parameters		
small.indel.bam.file	—	Input *.bam file(s). Multiple inputs are allowed and each are separated by a comma.

Table 56 Small indel .ini.file parameters *(continued)*

Parameter name	Default value	Description
cmap	—	CMAP file. See Appendix A, "File Format Descriptions" on page 295 for details on CMAP files.
Optional input parameters		
small.indel.combined.file	—	Intermediate (.pas.sum) file.
Outputs		
small.indel.candidate.dir	smallindel/	Results directory.
small.indel.output.prefix	indel-candidate-list-new	Filename suffix used for outputs.
Output options		
small.indel.detail.level	3	For BAM file inputs, the level of detail in output: • 9 is most detailed, but is slower. • 1-8 keeps only some of the alignment's anchor alignment but none of the ungapped alignment. • 0 keeps only position information about the anchor read and no information for the ungapped alignment.
sample.name		Places this identifier in the header of the .GFF, spaces escaped by \.
small.indel.zygosity.profile.name	max-mapping-2010-03-04	Zygosity profile name: • "classic-version-2009-10-16" for classic mapping. • "max-mapping-2010-03-04" for max mapping. • "all-homozygous" for all homozygous calls.
Pileup options		
small.indel.min.num.evid	2	Minimum number of evidences required for an indel call.
small.indel.max.num.evid	-1	Maximum number of evidences, use -1 for no maximum.
small.indel.consGroup	1	Indel grouping method: • 1 — conservative grouping with 5bp max between consecutive evidences • 0 — the higher of 15 or (7* indel size) maximum between evidences. • 9 — no grouping (options 1 and 9 are available for BAM files)
Mapping quality filtering		
small.indel.min.mapping.quality Note: requires BAM input	0	Keeps only reads that have this or higher pairing qualities. For paired tags, mapping quality is for the pair (pairing quality), and for fragment, it is the single tag's map quality.
small.indel.min.best.mapping.quality Note: requires BAM input	0	For a particular indel called with a set of reads, at least one pairing quality in this set must be higher than this value. Allows for supporting evidences to have a lower mapping quality threshold then the best read.

Table 56 Small indel .ini.file parameters *(continued)*

Parameter name	Default value	Description
small.indel.min.non.matched.length	10	Minimum non-mapped length for indel tag. Requires small.indel.detail.level=9.
small.indel.min.map.qv	-1	Minimum map QV for non-indel (anchor) tag. Requires small.indel.detail.level not be 0.
small.indel.min.map.length	-1	Minimum map length for non-indel (anchor) tag. Requires small.indel.detail.level not be 0.
Heuristic filtering		
small.indel.colorspace.compatibility.level	1	Colorspace compatibility level 0 — No color space filtering 1 — Require the most common indel allele has more reads with that allele than reads where the gap is not color space compatible.
small.indel.max.coverage.ratio	12	Maximum clipped coverage/# non-redundant support ratio. -1 for no limit.
small.indel.norequire.called.indel.size	0	Indels require that 75% of the reads call the same indel size. Set this to 1 to remove this requirement.
small.indel.filter.off	0	Makes all alignment pileups (.pas.sum) as hits. All filtering downstream of making the pileups are turned off, but all filters upstream (for example, min.num.evid, min.map.qv) are still active.
small.indel.max.nonreds-4filt	2	Maximum number of non-redundant reads where read position filtering is applied.
small.indel.min.from.end.pos	9.1	Minimum number of base pairs from the end of the read.
small.indel.max.ave.read.pos		Maximum average read position for filtering.
Indel size filtering		
small.indel.min.insertion.size		Minimum insertion size to include.
small.indel.min.deletion.size		Minimum deletion size to include.
small.indel.max.insertion.size		Maximum insertion size to include.
small.indel.max.deletion.size		Maximum deletion size to include.
Other parameters		
memory.request	3800	Memory request for process in mb.
small.indel.log.dir	smallindel-log-dir/	Tool log directory.

Prepare to run the Find Small InDels tool

Select the required input files

Before you can run the Find Small InDels tool you must know:

- The absolute path to at least one *.bam file.
- The absolute path to the *.cmap file.

- The Output Prefix name.

Complete the prerequisites

1. Complete the applicable prerequisites described in Chapter 3, "Before you Begin" on page 35.
2. Login to the BioScope™ Software cluster. Change to the working directory and update the small.indel.ini file with information that applies to the small indel run that you want to initiate. See "small.indel.ini file example" on page 277.
3. Complete the Mate Pair mapping process on the primary data from the instrument.

Run the Find Small InDels tool from the command line

Although several different software programs are involved in the run, a single command generates all of the related programs required to complete the run. The *.plan file that is specified in the command syntax controls the order in which BioScope™ Software runs the related programs.

Start the run

1. Connect to the BioScope™ Software cluster and login with a user ID that has write privileges on all of the directories that the BioScope™ Software uses when the tool runs.
2. At a command prompt, enter:
   ```
   bioscope.sh -l filename.log filename.plan
   ```

Do not log out of the BioScope™ Software cluster.

Check the run status from the command line

1. Navigate to the log directory that is defined in the small.indel.ini file. For example, you might enter:
   ```
   cd /data/results/tertiary/log
   ```
2. Open `bioscope.`*`yyyymmddhhmmss`*`.log`.
3. Scroll to the end of the file.

The run is complete if you see an entry similar to:

```
15 Apr 2010 03:16:32,537  INFO [main] PluginJobManager:130 -
>>>> END of PluginJobManager >>>> date DURATION=4 minutes 33
secs

15 Apr 2010 03:16:32,537  INFO [main] EventTransportFactory:129
- Closing JMS connection and session
```

Run the Find Small InDels tool from the web interface

The instructions in this section assume the following system conditions:

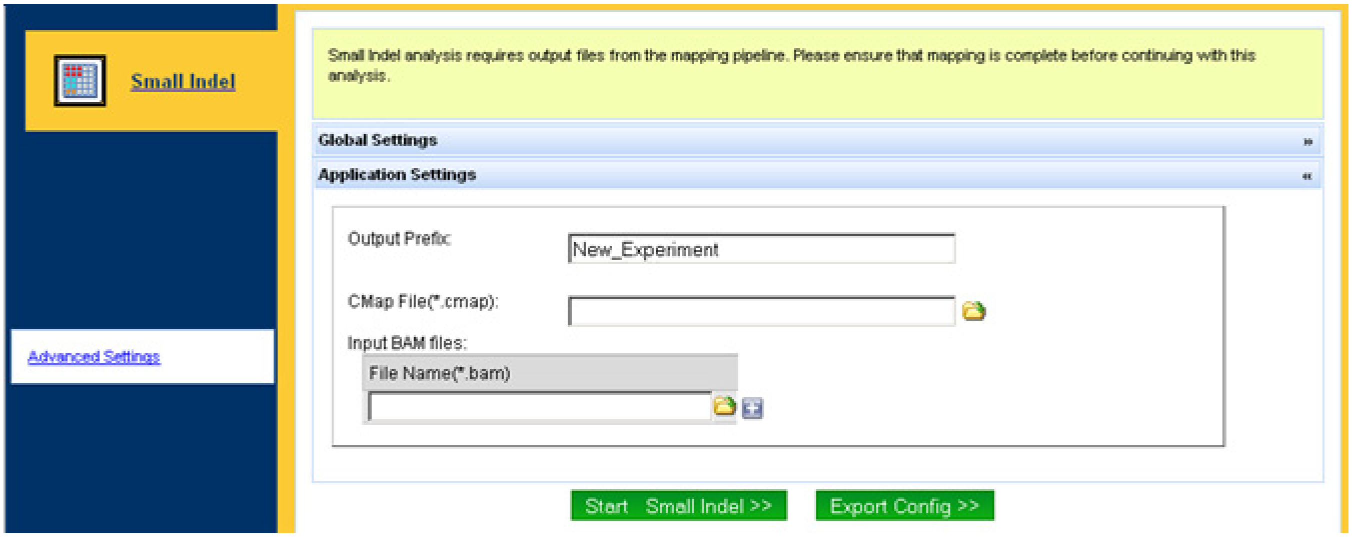
- The Java Messenger, Tomcat, and Apache services are running on the BioScope™ Software cluster.
- You are using Internet Explorer versions 6 or 7 or Mozilla 3.0.1.

- You have planned the name of the small.indel output file prefix.
- Mate-pair mapping is complete.

1. Launch a browser and enter the BioScope™ Software URL: http://<hostname>:8080/bioscope
2. Click **Find Small InDels**.

The Find Small InDels page has two windows and one link (see Figure 96).

- Global Settings
- Applications Settings
- Advanced Settings



Small Indel analysis requires output files from the mapping pipeline. Please ensure that mapping is complete before continuing with this analysis.

Output Prefix: New_Experiment

CMap File(*.cmap):

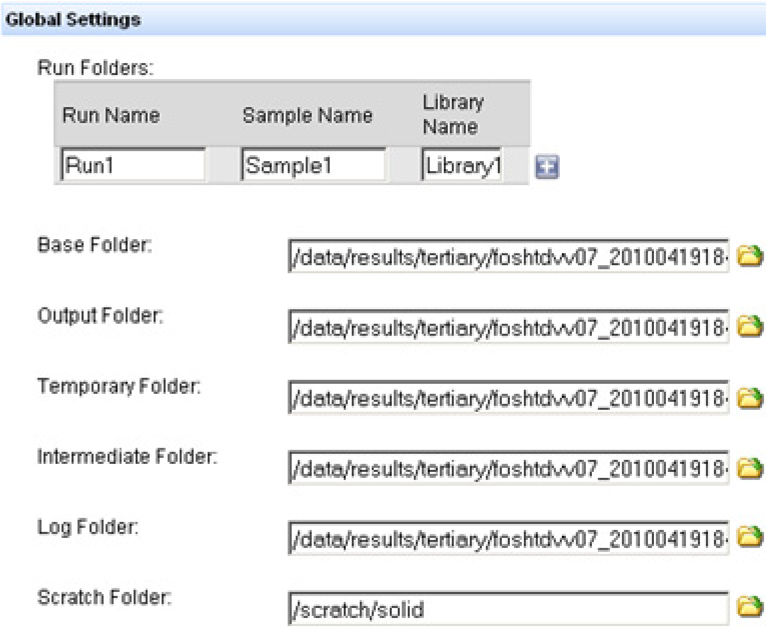
Input BAM files:

File Name(*.bam)

Figure 96 Find Small Indel page example

Global Settings description

The Global Settings window displays the default values for the folders that BioScope™ Software creates for the files that result from the Find Small Indels run (see Figure 97). The window also has fields where you can change default values for the Run Name, Sample Name, and Library Name of the primary data that was exported to BioScope™ Software from the instrument.



Global Settings

Run Folders:

Run Name	Sample Name	Library Name
Run1	Sample1	Library1

Base Folder: /data/results/tertiary/foshtdvw07_2010041918

Output Folder: /data/results/tertiary/foshtdvw07_2010041918

Temporary Folder: /data/results/tertiary/foshtdvw07_2010041918

Intermediate Folder: /data/results/tertiary/foshtdvw07_2010041918

Log Folder: /data/results/tertiary/foshtdvw07_2010041918

Scratch Folder: /scratch/solid

Figure 97 Find Small InDels Global Settings window example

Customize the default folder structure (optional)

The folders store the results files generated by each Find Small InDels run. BioScope™ Software automatically creates the default folder structure for each Find Small InDels run:
`/data/results/tertiary/`*`headnode_yyyymmddhhmmss_x`*

Complete the following steps to change the default directory structure.

1. Click 📂 in the Base Folder field. The File Browser dialog appears.
2. In the **Look in** field, type the custom directory path, for example, `/home/data`
3. Click **Open**.
4. The folders reflect the updated directory structure.

Note: If you change the default directory structure, the Output, Temporary, Intermediate, and Log folders become subdirectories of the Base Folder.

Update the Run Folder settings (optional)

You can accept the default values in the Run Name, Sample Name and Library Name fields. In this context, “run” refers to the primary data that was exported to BioScope™ Software from the instrument.

To change the default values for the Run Folders:

1. Enter the updated run name in the Run Name field.
2. Enter the updated sample name in the Sample Name field.

3. Enter the updated library name in the Library Name field.

4. Optional: Click ⊞ to add a row for a second run folder.

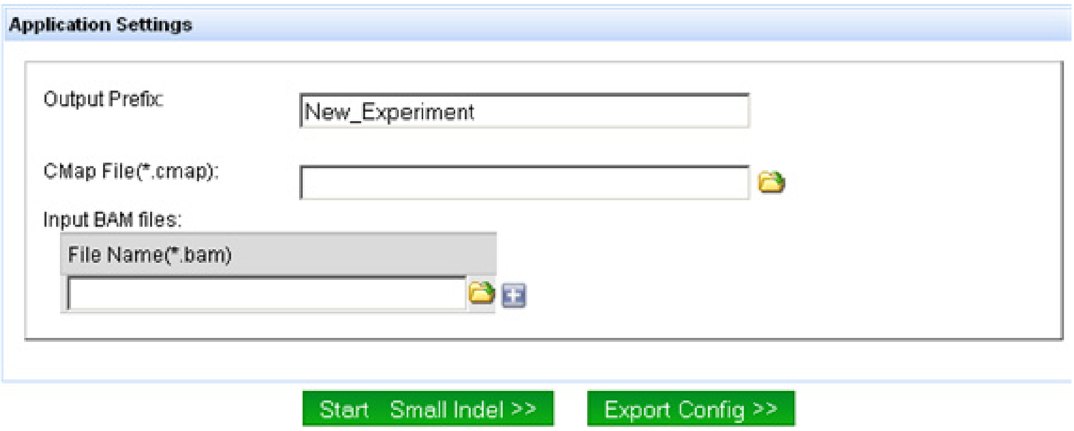
5. Optional: Enter a Run Name, a Sample Name and a Library Name in the new row.

Advanced Settings description

Click **Advanced Settings** to view the current default values defined by BioScope™ Software for the Find Small InDels tool. *Do not* change any Advanced Settings unless instructed to by the BioScope™ Software administrator.

Application Settings description

In the Application Settings window (see Figure 98 on page 283), you update the Output Prefix, define the absolute path to the *.cmap file, and you define at least one absolute path to a *bam file. You also start the Find Small InDels run from the Applications Setting section. The Export Config >> button is only used to process barcoded libraries (see Appendix C, "Batch Analysis of Barcoded Library Data" on page 319).



Application Settings

Output Prefix: New_Experiment

CMap File(*.cmap):

Input BAM files:

File Name(*.bam)

Figure 98 Find Small InDels Application Settings window

Start the Small InDels tool run

1. Update the Output Prefix or accept the default name.

2. Click 📂 in the CMap File(*.cmap) field. The File Browser window appears.

3. Define the absolute path to the *.cmap file.

4. Click **Open**.

5. Click 📂 in the FileName(*bam) field. The File Browser window appears.

6. Define the absolute path to the *.bam file.

7. Click **Open**.

8. Optional: Click ⊞ to define a path to a second folder that contains a *.bam file.

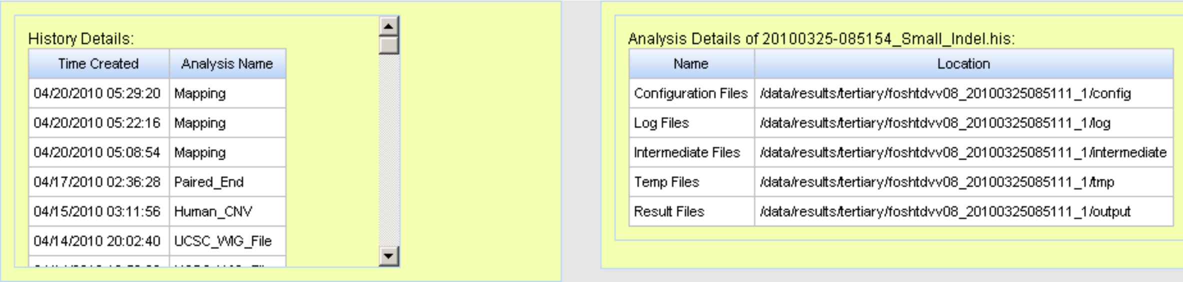
9. Click Start Small Indel >> to start the run.

10. At the job submission dialog, click **OK** after you have verified the folder locations.

Check the run status from the web interface

1. Click **History**. The History window appears. The History Details table is displayed in the left pane. The History Details table shows the Time Created and Analysis Name for all runs performed on the BioScope™ Software cluster.

2. Scroll the History Details table and select the Small_Indel run, based on the data in the Time Created column (see Figure 99).



History Details:

Time Created	Analysis Name
04/20/2010 05:29:20	Mapping
04/20/2010 05:22:16	Mapping
04/20/2010 05:08:54	Mapping
04/17/2010 02:36:28	Paired_End
04/15/2010 03:11:56	Human_CNV
04/14/2010 20:02:40	UCSC_WIG_File

Analysis Details of 20100325-085154_Small_Indel.his:

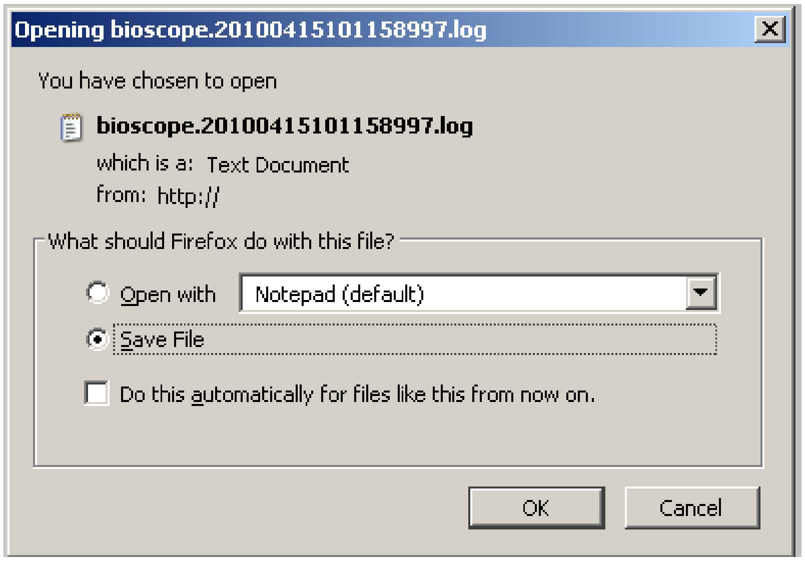
Name	Location
Configuration Files	/data/results/tertiary/foshtdvv08_2010032508511_1/config
Log Files	/data/results/tertiary/foshtdvv08_20100325085111_1/log
Intermediate Files	/data/results/tertiary/foshtdvv08_20100325085111_1/intermediate
Temp Files	/data/results/tertiary/foshtdvv08_20100325085111_1/tmp
Result Files	/data/results/tertiary/foshtdvv08_20100325085111_1/output

Figure 99 History details and analysis details for a Find Small InDels tool run

3. Double-click the Log Files row in the Analysis Details table. The File Browser dialog opens. Click **Resend** if your browser displays a message.

4. Select the `bioscope.`*`yyyymmddhhmmss`*`.log` file.

5. Click **Download**.
   - Click **Open with** and click **OK** to view the log file in Notepad or select a different text editor.
   - Click **Save File** to copy the file to your workstation.



Opening bioscope.20100415101158997.log

You have chosen to open

bioscope.20100415101158997.log

which is a: Text Document
from: http://

What should Firefox do with this file?

○ Open with Notepad (default)
● Save File

□ Do this automatically for files like this from now on.

Figure 100 Log file download page example

6. Scroll to the end of the file.

The run is complete if you see an entry similar to:

```
15 Apr 2010 03:16:32,537  INFO [main] PluginJobManager:130 -
>>>> END of PluginJobManager >>>> date DURATION=4 minutes 33
secs

15 Apr 2010 03:16:32,537  INFO [main] EventTransportFactory:129
- Closing JMS connection and session
```

Find Small Indels tool output file formats

Small indel GFF format

The main output file of the pipeline is the produced GFF_3 file. As shown in the small indel output file example below (after Table 57), the *.gff file created by the Small Indel tool begins with the General File Format Version 3 headers. The section following the GFF_3 headers displays BAM header and read group information. The lines containing information about each indel follows. Table 57 describes each column and attribute contained in the file. Table 58 on page 288 describes the *.txt format, which contains similar information as the GFF file.

The *.pas file (described in Appendix A, File Format Descriptions) is for legacy purposes and although produced, is not currently being used by the system. The *.pas.sum is the internal pileup format (see Figure 92 on page 267), and the *.align displays the alignments of reads of the gaps in text format. The *.ungapped file contains for each pileup, the list of beads ids from reads that are aligned without gaps but also span the location of the indel.

Table 57 Small indel GFF file format descriptions

Column/Attribute tag name	Description	Example
seqid	The ID of the sequence to which the start and end coordinates refer, such as a chromosome number.	chrV
source	Free-text qualifier that indicates the algorithm or method that generated the feature.	AB_SOLiD Small Indel Tool
type	SOFA feature. For indels, possible values are: • insertion_site • deletion	deletion
start/end	1-based integer coordinates of the feature, relative to the sequence in column 1. For zero-length features, such as insertion sites, "start" equals "end" and the implied site is to the right of the indicated base in the direction of the landmark. For deletions, the start and end indicate the positions in the reference that are not present in the sample.	200587060/200587063
score	Floating-point value representing the quality of the evidence for the feature. This is currently set to 1.	1
strand	The strand of the feature. The type of indels detected are not stranded, because they are found with sequence reads that are on either or both strands	.

Table 57 Small indel GFF file format descriptions *(continued)*

Column/Attribute tag name	Description	Example
phase	Translation frame; relevant only for CDS features.	.
Attributes		
ID	Unique indel id. Non-sequential ids are due to filtering.	21
ins_len	Number of bases inserted relative to the reference.	
del_len	Number of bases missing from the reference.	2
allele-call-pos	Position of the first base of the allele call. If the first allele is missing, it is the position immediately before the insertion.	713662
allele-call	The reference and the indel alleles. The first one is always the reference, even if zygosity is called HOMOZYGOUS.	ag/
allele-pos	Position of the first base of the alleles. If the first allele is missing, it is the position immediately before the insertion.	713660
alleles	A complete list of alleles found by all reads of the pileup, and the bases representing the possible short repeat around the indel. The first one is always the reference.	acagagagaag/ aCAGAGAAG
allele-counts	The number of indel reads found for each of the above alleles. The first allele is the reference.	REF,3
tight_chrom_pos	Conservative estimate of chromosome position range of the feature. This ambiguity is resolved by the allele-pos tag.	713662-713667
loose_chrom_pos	Estimate of the maximum chromosome position range of the feature. This ambiguity is resolved by the allele-pos tag.	713662-713667
no_nonred_reads	Number of reads with unique start positions (non-redundant reads).	3
coverage_ratio	Clipped normal coverage/number of non-redundant reads. Clipped coverage is where the parts of the read that are within 5 bp at either end are not counted as a part of coverage.	1.6667
experimental-zygosity	Experimental zygosity call.	HEMIZYGOUS
experimental-zygosity-score	Experimental zygosity score. It is not rigorously a p-value.	0.9656
run_names	Run names (one for each input) for each read. For BAM files, the 1 in 50_1_r is the runIdNum in the ##@HD header line of the gff.	L1_1_50_1_r, L1_1_50_1_r, L1_1_50_1_r

Table 57 Small indel GFF file format descriptions *(continued)*

Column/Attribute tag name	Description	Example
bead_ids	Bead IDs for each read	984_536_1054, 1431_2007_1567, 116_364_1582
overall_qvs	Alignment quality values for each bead.	26,47,66
no_mismatches	List of number of mismatches for each read.	-1,-1,-1
read_pos	Position in each non-redundant read at which the In/Del occurs.	20,17,10
from_end_pos	Same as above, except that the value is the number of base pairs from the end of the read.	30,33,40
strands	Strand for each read.	+,+,+
tags	Tags where the indel was found. Possible values are F3, R3, and FRAG.	F3,F3,F3
indel_sizes	List of sizes of indel found for each evidence.	-2,-2,-2
non_indel_ no_mismatches	Number of mismatches of the other tag that was matched without a gap. Values of NIL occur if that particular bead is from a fragment library.	1,3,3
unmatched-lengths	For a particular bead-id, the length of the read that was left unmatched (equal to the read length if no ungapped match was found). This is relevant in extended read alignments.	50,50,50
ave-unmatched	Average of the unmatched lengths.	50
anchor-match-lengths	Anchor tag's match lengths in extended read alignments.	49,44,49
ave-anchor-length	Average of the anchor match lengths.	47.3333
read_seqs	The read sequence where the indel was found for each bead.	T302000, T120320, T232132
base_qvs	The color call QVs for the read sequence where the indel was found. This is not displayed by default.	—
non_indel_seqs	Sequences of the non-indel anchor tag.	G100220, G313000, G203330
non_indel_qvs	The color call QVs for the non-indel anchor tag sequence.	—

An example of a small indel output file is shown below. The file contents are as follows:

1. The general GFF version 3 headers.
2. The BAM header and read group information.
3. Lines contain information about each indel.

Table 57 on page 285 describes each tag of the attributes column.

```
##gff-version 3
##solid-gff-version 0.3
##source-version SOLiD small-indel-tool.pl/process-small-indels v 1.3.1, 2010-04-09 14:51:59
```

```
##type DNA
##date 2010-04-18
##time 03:28:56
##feature-ontology http://song.cvs.sourceforge.net/*checkout*/song/ontology/
sofa.obo?revision=1.141
##reference-file
##input-files /data/results/instName_runName/outputs/pairing/F3-R3-Paired.bam
##run-path /data/results/instName_runName/workdir/small-indels
##Filter-settings: max-ave-read-pos=none,min-ave-from-end-pos=9.1,max-nonreds-4filt=2,min-
insertion-size=none,min-deletion-size=none,max-insertion-size=none,max-deletion-
size=none,require-called-indel-size?=T,max-coverage-ratio=12,min-mapping-quality=none,min-
best-mapping-quality=none
##BAM header:
##@HD   VN:1.0  SO:2    runIdNum:1
##@RG   ID:20100319024304802    SM:HuRef        LB:50x50MP      PU:bioscope-pairing
PI:1575 DT:2010-03-18T19        PL:SOLiD
chr1    AB_SOLiD Small Indel Tool       deletion        713662  713663  1       .       .
ID=21;del_len=2;allele-call-pos=713662;allele-call=ag/;allele-
pos=713660;alleles=acagagagaag/aCAGAGAAG;allele-counts=REF,3;tight_chrom_pos=713662-
713667;loose_chrom_pos=713662-
713667;no_nonred_reads=3;coverage_ratio=1.6667;zygosity=HEMIZYGOUS;zygosity-
score=0.9656;run_names=L1_1_50_1_r,L1_1_50_1_r,L1_1_50_1_r;bead_ids=984_536_1054,1431_2007_1
567,116_364_1582;overall_qvs=26,47,66;no_mismatches=-1,-1,-
1;read_pos=20,17,10;from_end_pos=30,33,40;strands=+,+,+;tags=F3,F3,F3;indel_sizes=-2,-2,-
2;non_indel_no_mismatches=1,3,3;unmatched-lengths=50,50,50;ave-unmatched=50.0000;anchor-
match-lengths=49,44,49;ave-anchor-
length=47.3333;read_seqs=T30202022221323122222112222020222222202213100002000,T12022221323122
2221122220202222222200213110002201320,T23231222221122220202222222002131100022013203120132;bas
e_qvs=;non_indel_seqs=G10001322230022300330033003300330000130130021200220,G31333120300212203
0110000202330022110000000013000000,G20331023330120011.000013222.0022.003300330033003 30;non_in
del_qvs=
```

Small indel TXT format

Table 58 describes the format of small indel txt files

Table 58 Small indel TXT file format description

File Name/Column	Description	Example
chrom	Chromosome number of indel.	1
min-chrom-pos	Start position of the indel.	713662
max-chrom-pos	End position of the indel	713663
called-range	Range of chromosome position range of the feature. Note: This ambiguity is resolved by the allele-call-pos tag in the gff.	713661-713661
tight-range	Conservative estimate of chromosome position range of the feature.	713662-713667
loose-range	Estimate of the maximum chromosome position range of the feature.	713662-713667
num-pos-strand	Number of reads that were mapped to the positive strand.	3
num-neg-strand	Number of reads that were mapped to the negative strand.	0
num-r3-hits	Number of reads where the indel was found on the R3 or F5 tag.	0
num-f3-hits	Number of reads where the indel was found in the F3 tag.	3
num-frag-hits	Number of reads where the indel was found from a Fragment tag.	0

Table 58 Small indel TXT file format description *(continued)*

File Name/Column	Description	Example
indel-type	INSERTION, DELETION, or COMBINATION. Combination is where there was no called indel size, and there were reads indicating both an insertion and a deletion.	DELETION
unique-indel-size	Called indel size.	-2
indel-size range	The indel sizes reported from each of the reads.	-2 to -2
num-uniq-align	Number of non-redundant alignments in the pileup.	3
num-tot-align	Total number of reads in the pileup.	3
average-read-position	Average read position where the gap occurred.	15.6667
ave-from-end-read-position	Average from end read position where the gap occurred.	34.3333
indel-read-pos-list	List of read positions where gap occurred.	20;17;10
dbsnp-indel	Reserved field for comparison information, such as comparison with dbSNP.	17
uw-hgsv-indel	Reserved field for comparison information, such as comparison with dbSNP.	24.5
read-lengths	Lengths of the reads (full read sequence, not the extended match length).	50;50;50
paired-distances	The clone sizes of each of the bead pairs (NIL for fragments).	1426;1370;1454
ave-pair-dist	Average of the paired distances.	1416.667
tags-R3-F3	Tags where the indel was found. Possible values are F3, R3, and FRAG. Note: for PE data, R3 represents the F5 tag.	F3;F3;F3
chrom-pos-s	Chromosome positions of the indel tag's match location.	713641;713644;713651
strands	Strand for each bead id.	+;+;+
indel-sizes	List of sizes of indel found for each read.	-2;-2;-2
nums-mismatches	List of number of mismatches for each read.	-1;-1;-1
ave-numb-mismatches	Average of the number of mismatches found in the indel tag.	-1
indel-lower-pos-s	List of lower ranges of the indel.	20;17;10
indel-upper-pos-s	List of upper ranges of the Indel. The lower and upper ranges represent the ambiguity of the gap alignment, for example, AT/-- in the context of ATAT/AT).	25;22;15
run-names	Run names (one for each input file) for each read. For BAM files, the 1 in 50_1_r is the runIdNum in the ##@HD header line of the gff.	L1_1_50_1_r;L1_1_50_1_r;L1_1_50_1_r
clone-ids	Bead IDs for each read	984_536_1054;1431_2007_1567;116_364_1582
read-seqs	The read sequences for each read.	T302000;T120320;T232132
ref-allele	Reference allele.	acagagagaag
var-allele1	Most common variant allele (if it passes the color error correction, otherwise it is NULL).	aCAGAGAAG

Table 58 Small indel TXT file format description *(continued)*

File Name/Column	Description	Example
other-var-alleles	Other alleles of the variant, if present.	—
var-counts1	Number of reads that have the most common indel allele.	3
other-var-counts	The list of the other allele counts.	NO_CALL
ungapped-unmatched-lengths	The unmatched length, unmatched lengths for each bead id. If the bead matches it is read length minus the extend length. If it does not, than it is the read length.	50,50,50
ave-ungapped-unmatched	Average of the ungapped, unmatched lengths.	50
anchor-match-lengths	Anchor tags match length (if it's classic, this will be the read length).	49,44,49
ave-anchor-length	The average of the anchor match lengths.	47.3333
coverage-ratios	Clipped normal coverage/number of non-redundant reads.	1.6667
zygosity-call	Experimental zygosity call.	HEMIZYGOUS
zygosity-p-value	Experimental zygosity score. It is not rigorously a p-value.	0.9656

16 Run ChIP-Seq

This chapter covers:

About ChIP-Seq data analysis tools

BioScope™ Software provides the ability to map data and create an output file type compatible with a variety of third-party Chromatin Immunoprecipitation Sequencing (ChIP)-Seq data analysis tools. The ChIP-Seq application has publicly available analysis software that can be used with BioScope™ Software output.

The ChIP assay is a method for analyzing epigenetic modifications and genomic DNA sequences bound to specific regulatory proteins. ChIP-Seq is a combined assay and sequencing technique for identifying and characterizing elements in protein-DNA interactions. It typically examines transcription factors (TF) bound to DNA and finds DNA sequence motifs common to binding sites.

Using the MAGnify™ ChIP-Seq kit with the SOLiD™ sequencing system enables you to generate sequence read data from a ChIP-Seq experimental approach. BioScope™ Software gives you the option to map the read data. The ChIP-Seq tool can only be used through the GUI.

Run ChIP-Seq

The ChIP-Seq tool performs the resequencing mapping processes. To map ChIP-Seq data, follow the instructions in Chapter 9 on page 117 and Chapter 10 on page 149.

After the mapping steps are complete, the resulting *.bam file can be used with compatible third-party commercial and academic ChIP-Seq analysis software tools (see Figure 101 on page 293).

As of this writing, you can download a BAM-to-BED format converter from third-party tools sites, for example:

code.google.com/p/bedtools

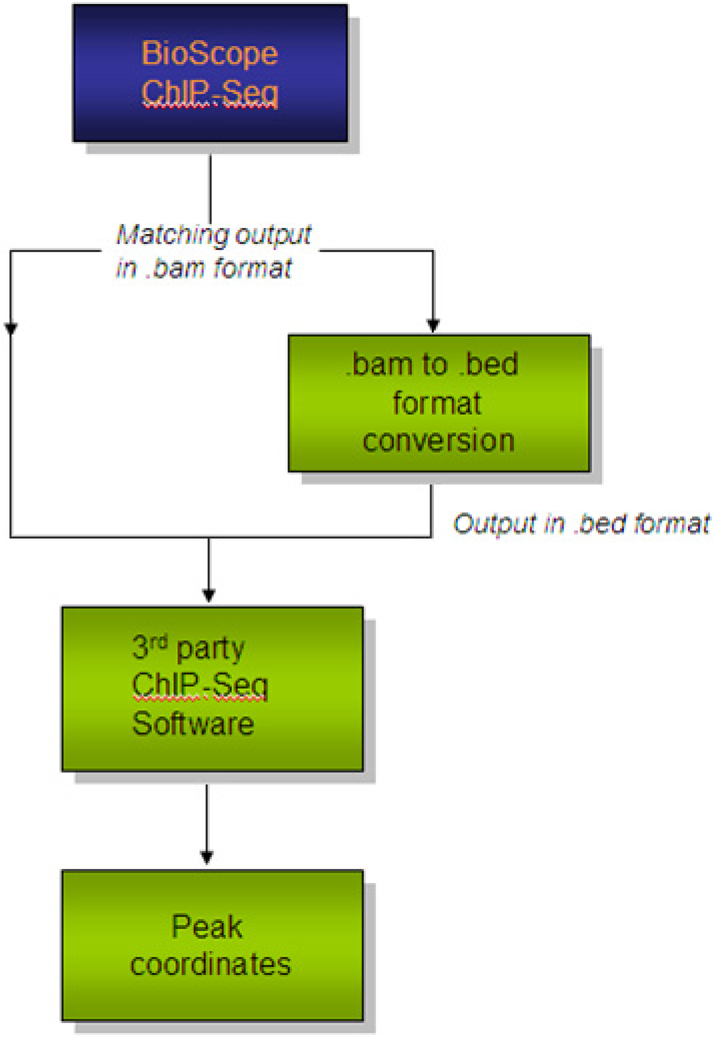

BioScope ChIP-Seq
Matching output in .bam format
.bam to .bed format conversion
Output in .bed format
3rd party ChIP-Seq Software
Peak coordinates

Figure 101 Compatible ChIP-Seq analysis software tools

A

File Format Descriptions

This appendix covers:

Introduction

This section describes specific information about the SOLiD™ *.bam file contents. You should be familiar with the general SAM specification and with the SAM specification field definitions.

Before SOLiD™ 4.0, the primary SOLiD™ read and alignment format was based on the public GFF specification. The specification defined the location of an aligned read on a reference sequence and provided for arbitrary name-value pair attributes. For the SOLiD™ GFF, attributes were used to detail color reads and quality values, base space translations, mate-pair information, and other aspects of the aligned read. Increased throughput of second-generation sequencing technologies resulted in the definition of the SAM file format and a compressed, indexed binary format (*.bam) that expands the basic alignment information of a *.gff file to include paired-read information and a more structured attribute set. A number of tools exist for basic manipulation of *.bam files, and an array of viewers and analysis tools are available.

BioScope™ Software secondary analysis (mapping and pairing) now produces a *.bam file as the main alignment format. Mate-pair and paired-end analysis directly produces a *.bam file, while a single file conversion is needed for fragment libraries (see Figure 102 on page 296). Depending on the output filter selected, unmapped and secondary alignments can be included.

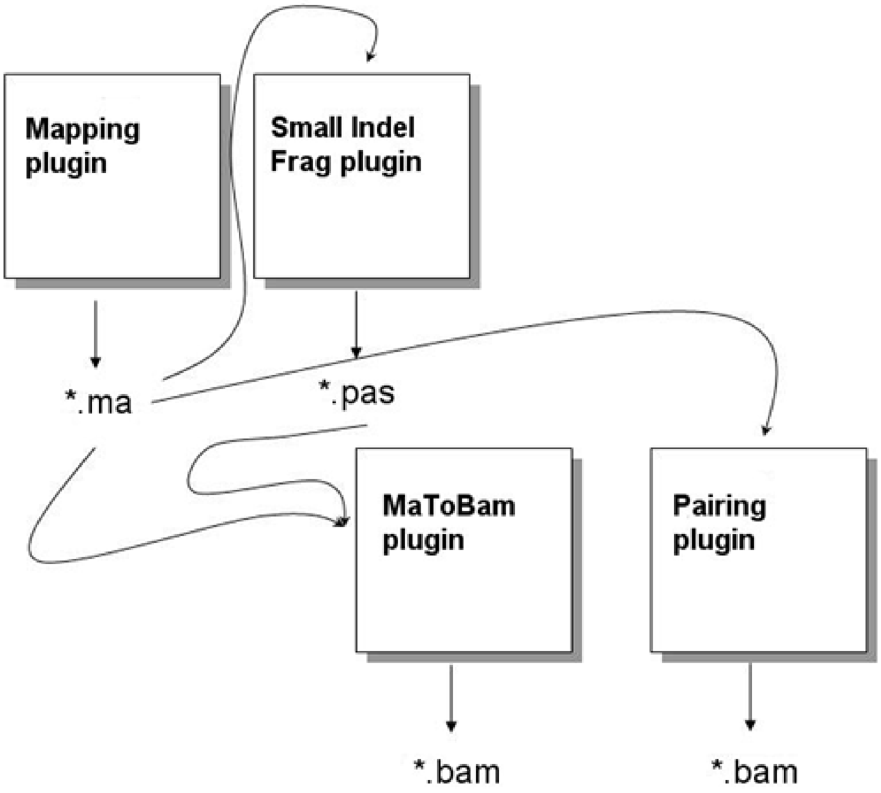


Figure 102 Diagram showing the creation of *.bam files from pairing and MaToBam tools

All tertiary analysis tools, such as Inversion or CNV, accept SOLiD™ *.bam files as input. Therefore, in order to perform downstream analysis, a *.bam file must be generated.

The *.bam files created by the MaToBam and Pairing tools are generated in coordinate order. Downstream analyses typically process data by position and generating *.bam files in coordinate order also allows chromosome-by-chromosome parallel processing. If query name order is needed, use a third-party tools, for example, `samtools sort`, to process query name order.

Content options

The MaToBam plugin supports content subsets that are similar to the MaToGff "convert" options in the 3.* releases. As a result, users have the ability to balance the information needed for downstream analysis against the size of the output. The output is controlled by the `ma.to.bam.output.filter` key. The possible values are described in Table 59.

Table 59 Output filter keys

Parameter	Description
primary	Reports only the primary alignment. Each bead id has a single primary alignment that corresponds to the highest mapping quality value. If multiple alignments have the same highest value one is randomly selected. Unmapped reads are placed in a separate file. All indel alignments are included.
alignment_score	• For reads with a single hit, a single corresponding BAM entry with mapping quality is reported. • For reads that have more than one hit, let s1 be the hits' highest local alignment score, and s2 be their second highest. Then alignment1 is deemed to map uniquely if s1 −s2 > cz, where cz is the clear zone defined by the ma.to.bam.clear.zone option (see Table 21 on page 140). If it is unique, alignment1 is reported in the BAM file with positive mapping quality. • For reads that have more than one hit that are not unique by the clear zone definition, the highest scoring alignment is reported, with mapping quality set to zero.
none	No filtering is done. All alignments are reported.

Carefully choose your content options so that the appropriate subset is selected for the intended downstream tool. If too stringent a filter is selected, the downstream tool may not have sufficient information to perform an analysis. If too loose a filter is selected, a downstream analysis may have to wade through a large amount of irrelevant data. If many analyses are to be performed, the lowest common denominator should be used.

Header details

The BAM file generates all of the header information required by the SAM format specification, including @HD, @SQ, and @RG lines.

To view the content of the BAM file header, use the following command:

```
samtools view -H <BAMfilename>
```

Sequence dictionary (@SQ)

Sequence header lines include the reference file URL, for example, `file:///share/reference/genomes/hg18.fa` in the optional UR field of the reference file. Downstream tools that need reference information can use the URL to reduce the number of user options. This value might become invalid if you relocate files.

Read group (@RG)

Read groups receive an arbitrary ID and sample name. The library field (LB) contains information that is important to downstream algorithms that use pairing information. A library name, which is specified by the tool parameter library.name, and the library type, are separated by a dash in the LB field. The library type is a structured value that details the nominal length of the two tags and the protocol used. as shown in the following syntax example:

```
l1(xl2)[F|MP|RR|RRBC]
```

In the syntax example, *l1* is the nominal length of the first read and *l2* is the nominal length of the second read. There will only be one number for fragment libraries. The library types correspond to fragment (*F*), mate-pair (*MP*), reverse read (*RR*), and reverse read-bar coded (*RRBC*).

Detecting structural variations, particularly large insertions and deletions, depends on the statistically likely range of pairing insert (PI) sizes. The pairing tool generates the information about PI sizes. The information has been used in legacy file formats to define the three-letter pairing "category", specifically the third letter. The PI field in the read group captures the range of pairing insert sizes with a range of the form shown in the following example:

```
PI;low-high
```

In the PI example, low is the lower bound of the pairing range, and high is the upper bound.

Color-space specifics

Color attributes

The SAM format specification includes the attribute tags CS, CQ and CM. All *.bam files support color-space reads (see Table 60).

Table 60 Color attribute tag description

Attribute tag	Description
CS	Color-space (CS) read. The CS field contains the original color-space read, which includes the primer base, in the orientation of the *.csfasta file. CS entries are not manipulated to be top-strand relative.
CQ	Color qualities. Color qualities are encoded according to the ascii-33 scheme used for the QUAL field. The orientation is the same as the orientation used for the *.csfasta file.
CM	The number of color-space mismatches.

Hard clipping of incomplete extensions

BioScope™ Software mapping uses a seed-extend algorithm. The algorithm increases mapping throughput by matching a seed, usually 25 bp, and extending the alignment until mismatches drive down the alignment score. Many alignments do not completely cover the color-space read. Because the base space sequence of color reads cannot be precisely known in the absence of alignment, incomplete extensions are represented as a hard-clip (H) operation in the *.bam CIGAR string (see Figure 103).

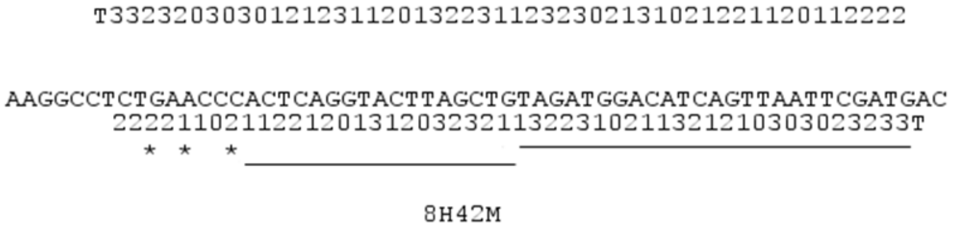

T33232030301212311201322311232302131021221120112222
AAGGCCTCTGAACCCACTCAGGTACTTAGCTGTAGATGGACATCAGTTAATTCGATGAC
22221102112212013120323211322310211321210303023233T
* * *
8H42M

Figure 103 Example of hard clipping from a color alignment

Figure 34 shows the read in normal orientation (see the top section) and aligned in reverse orientation to the reference top strand (see the middle section). The lines below the alignment show the extent of the seed (top horizontal line) and extension (bottom horizontal line) phases of mapping. The extension only results in 42 bases of alignment. The remaining portion of the color alignment has a number of mismatches that prevent extension. These are coded as hard-clipped regions. The CIGAR field in the *.bam file is top-strand relative, so even though the hard clipping is on the end of the reversed color read, it is on the beginning of the CIGAR string.

Visualizing *.bam output

You can use third-party software visualization tools to view *.bam files in a browser.

Integrative Genomics View (IGV)

The Integrative Genomics Viewer (IGV) available from the Broad Institute is a visualization tool for interactive exploration of large, integrated datasets. The IGV reads *.bam files directly, which allows for easy viewing and inspection of alignments against the genome (see Figure 104).

For more information, go to **www.broadinstitute.org/igv/**

If you use IGV to visualize the *.bam files, verify that the *.bai file is present. The *.bai file is the index that is built for *.bam files and is a standard part of the public SAM specification. If the pairing and MaToBam tools do not automatically create the *.bai file:

1. Login to the BioScope™ Software cluster.
2. At a command prompt, enter:

```
$ samtools index <bam file name>.bam
```

Indexing only works if the file is sorted in "coordinate" order. If the file is not sorted in “coordinate order”, login to BioScope™ Software. At a command prompt, run the following command to sort the file in coordinate order:

```
$ samtools sort <unsorted bam name>.bam <sorted bam name>
```

UC Santa Cruz (UCSC) genome browser

The UCSC Genome Browser serves as an interactive web-based "microscope" that allows researchers to view all 23 chromosomes of the human genome at any scale, from a full chromosome down to an individual nucleotide.

For more information, go to the Genome Browser web site:
www.cbse.ucsc.edu/research/browser

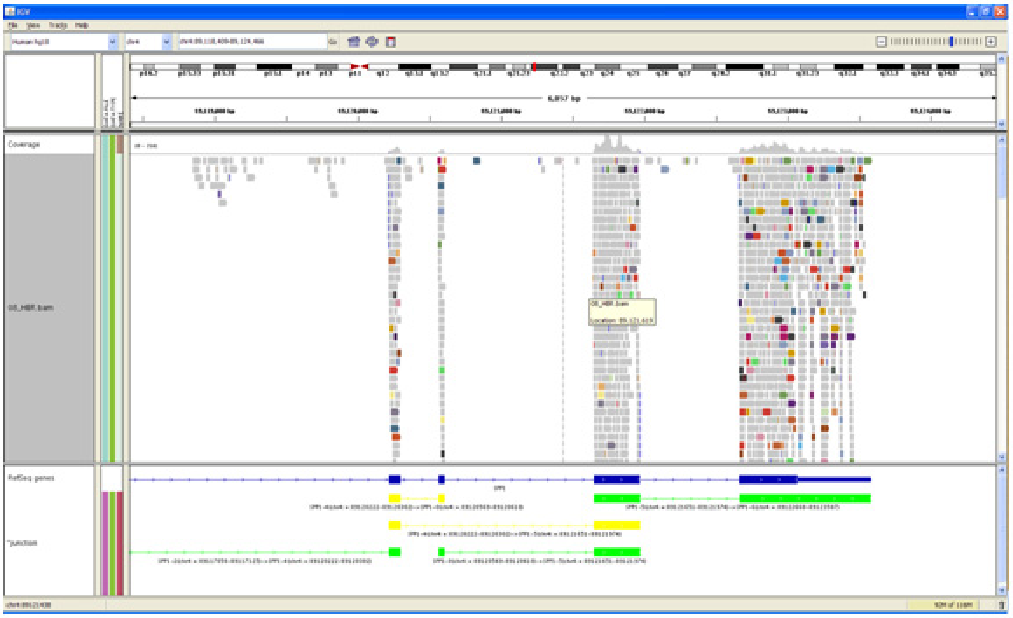


Figure 104 An example of a *.bam file visualized in the IGV viewer

Pairing information in a *.bam file

The *.bam file that is produced by the pairing tool supports both mate-pair and paired-end protocols using the standard SAM format fields, in particular the ISIZE and FLAG fields.

Calculation of tag names

The paired libraries use tag names to refer to members of the pair. The mate-pair libraries use F3 and R3 as the tag names. The paired-end libraries use F3 and F5 as the tag names. Use the FLAG field and information from the LB field of the read group to recapitulate tag names (see Table 61).

Table 61 Calculating tag names

FLAG bit	Library type	Tag name
0x0040 (first read in a pair)	MP	F3
0x0080 (second read in a pair)	MP	R3
0x0040	RR	F3
0X0080	RR	F5
0X040	F	F3

Proper pairs

Legacy file formats, such as *.mates, and *.gff, described pairs using a three-letter category. Pairs in the AAA category correspond to the “proper pair” concept in the SAM format. The pairs reflect pairings that are not altered by a structural variation such as an inversion or deletion (see Figure 105). The *.bam file field values for proper pairs are different for mate-pair and paired-end libraries:

Mate-pair libraries

- Strand flag is equal for both mates (both 0 or both 1).
- ISIZE is between the lower and upper limit of the insert range.
- For forward strand hits R3 POS < F3 POS.
- For reverse strand hits F3 POS < R3 POS.

Paired-end

- Strand flag is opposite for the mates.
- ISIZE is between the lower and upper limit of the insert range. In the case of paired-end libraries, the ISIZE might be smaller than the sum of the alignment lengths.
- F3 POS < F5 POS if F3 is on the forward strand.
- F5 POS < F3 POS if F5 is on the forward strand.

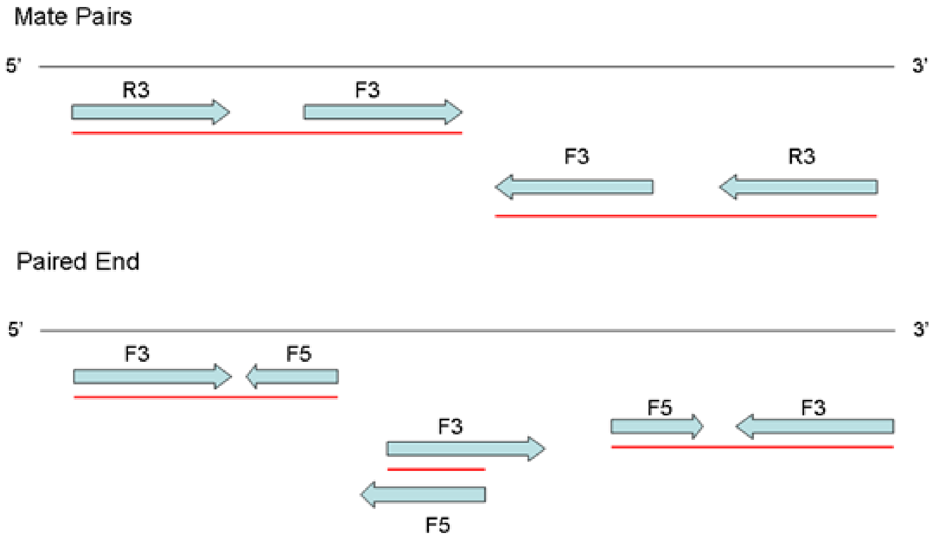


Figure 105 Example of proper pairs for mate-pair and paired-end protocols

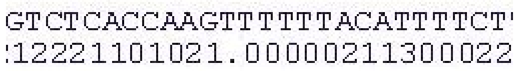
Single read mapping quality

As described in the SAM format specification, the MAPQ field for paired results contains a pairing quality value. Under some circumstances. it is valuable to include the original single-read alignment quality value. The original single-read alignment value is maintained in the SM:i attribute in *.bam files.

Indel alignments

Indel-containing alignments are no longer processed by downstream tools via the *.pas file format. The alignments are now included in the secondary analysis *.bam file. Indel alignments are included in quality calculations and primary alignment/pair selections.

Nearly all of the fields in the *.pas file are represented in SAM format. However, there is no allowance for ambiguous locations in the CIGAR string or elsewhere. Ambiguity occurs when a repeat element is inserted or deleted with respect to the reference sequence. Indel alignments include a user-defined attribute (XW:Z) to specify the range of possible locations within the read for an indel in a repeat region (see Figure 106).



```
GTCTCACCAAGTTTTTTACATTTTCT'
:12221101021.00000211300022
```

Figure 106 Example indel in a repeat region with ambiguous placement

Referring to Figure 106, in the alignment shown of a reference sequence stretch with a color read, the deletion of a single T, which is represented as a dot in the color sequence, cannot be placed precisely because of the repeated Ts. The XW attribute would span the homopolymer region (30_35).

Table 62 PAS file column descriptions

Field Name	Example	Description
Genome position	4294973387	The genome position given by this formula: C * 232 + P – 1 where *C* is the chromosome number and *P* is the position on that chromosome.
Indel size	-7	Number of bases in the indel. A negative value means a deletion, a positive value is an insertion, for example, -8 is a deletion of size 8, and 3 is an insertion of size 3.
Number of errors	2	Number of errors in the tag where the indel was found.
Alignment	See "PAS format example" below.	Details of the alignment in the form of a number of concatenated fields.

The PAS file contains four tab separated columns as described in Table 62. The fourth column contains the full alignment information. One example of an alignment is illustrated as follows.

PAS format example

The following is an example of PAS file content:

```
>600_16_579_14_Lib1_1_50_1_runName_sampleName,1_6074.51.2(17:17
_17)[G20]| 1_7297.1:(44.8.0)[T02] !8B52/
:B;*%=7+'539&)4>455++60+0<9.(2
```

The alignment is in the form of a number of concatenated fields. The text represents:

<data source>,<R3 tag details>|<F3 tag details> R3_FASTQ F3_FASTQ

where the indel may be either F3 or R3, and the other tag would be the anchor, ungapped alignment. The R3_FASTQ and F3_FASTQ is the color quality strings for those tags in FASTQ base quality format. For fragment, there is only one tag:

<data source>,<tag details> FASTQ

Table 63 explains the fields in the example PAS file content.

Table 63 PAS file field descriptions

Field	Meaning	Notes
Data source fields		
`600_16_579`	The bead ID.	
`14`	the unmatched length for that bead if an ungapped extended alignment was found for that bead.	The value `99999` is used if there was no ungapped alignment found.
`Lib1_1`	A library indicator.	Deprecated.
`50`	The full read-length.	
`1`	The run ID.	

Table 63 PAS file field descriptions *(continued)*

Field	Meaning	Notes
`runName_sampleName`	The concatenation of the run name and the sample name.	
Indel tag fields		
`1_6074.26.2`	A sequence `1` hit at position `6074`. `26` is the match length with `2` mismatches.	
`(17:17_17)`	The read position found was position `17` (zero-based). `17_17` is the range of other possible positions. `[G20]` is the read sequence with the gap alignment.	insertion size = read_length - match_length – 1 Negative values are deletions, and positive values are insertions. In this example, the read length was 50, so the indel is a deletion of 2. If an indel is not detected in a tag, the Tag indel field contains the mapping hit information: <seq index>_<alignment start>.<num mismatches>
Anchor (non-indel) tag fields		
`1_7297.1:(44.8.0)[T02]`	The alignment and the sequence of the anchor non-indel tag. `1:` the chromosome index (not necessarily the chromosome number). `7297:` the chromosome position of the alignment. `1:` the number of mismatches in the anchor part of the alignment. `44:` the alignment extended length. `8:` the overall number of mismatches. `0:` the start of the alignment. `[T02]:` the color space sequence.	The alignment is the same format found in the .ma file except that it is followed by the color space sequence (`[T02]`).

Note: The positions in the PAS files are 0-based. This contrasts with the positions reported in the TXT/GFF files, which have been adjusted to be 1-based.

Whole transcriptome output

See Table 64 for information about custom attributes.

Table 64 Custom attributes

Attribute key	Description
XL:z	This key contains a comma-separated list of ints. The key also stores the color-space length of all alignments containing the current query. The alignments are sorted by genomic location.

Table 64 Custom attributes *(continued)*

Attribute key	Description
XU:z	This key provides a comma-separated list of ints. The key also provides the number of color mismatches in alignments containing the current query. The alignments are sorted by genomic location.
XF:z	"T" for true or "F" or false. "T" is used if the read is filtered.
XJ:z	"K" for Known putative junction, or "P" for putative junction.

Legacy format translation

Table 65 and Table 66 map the *.gff file field name to the corresponding *.bam file field name for header fields and alignment data.

Table 65 *.gff file header fields

GFF field	Corresponding *.bam field	Comments
##reference-name	@SQ UR field	This header field contains the reference file name in the UR field.
##history	—	—
##color-code	—	Only a single color encoding is used.
##primer-base	—	This field is not needed. The primer base is stored with the color read attribute.
##max-num-mismatches	—	—
##max-read-length	@RG LB field	The read lengths in the file are determined from the library type information in the LB field of the read group.
##line-order	@HD SO field	The sort order field in the *.bam header indicates the line order. Valid values are: • None • Coordinate • Qname

Table 66 Alignment data

GFF field	Corresponding *.bam field	Comments
seqid	RNAME	The *.gff files used an ordinal number to indicate the reference contig. The ordinal number could be translated to names by a header table. The *.bam uses the @SQ lines to perform a similar lookup.
source	@RG PL	The read group platform (@RG PL field) is used in a manner similar to the SOLiD™ *.gff source.
type	—	All records are alignments.
start	POS	The 1-based, left-most, top-strand position of the alignment.

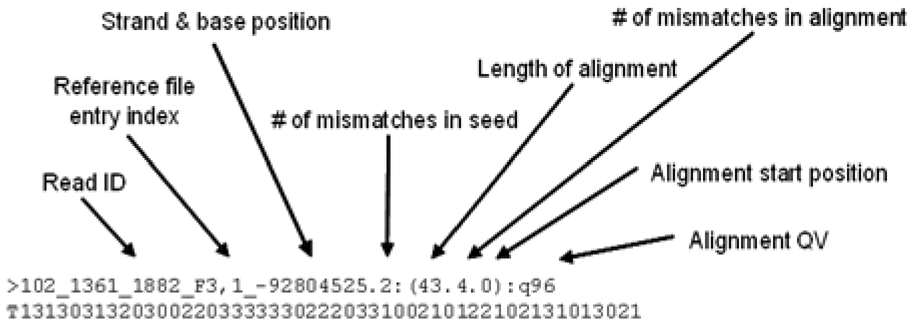
GFF field	Corresponding *.bam field	Comments
end	Calculated from CIGAR	The *.bam file does not support an explicit end point. An explicit end point can be calculated from the CIGAR string.
score	—	A read quality is not stored in the *.bam alignment record. Mapping and pairing quality is stored in the *.bam alignment record.
strand	5th FLAG bit (query strand)	The fifth bit of the FLAG field indicates the strand of the alignment.
GFF attributes		
aID (*bead id*)	QNAME	The bead id.
at (*adaptor type*)	Calculated value	The adaptor type or primer set field can be calculated as described in "Calculation of tag names" on page 301.
b (*bases*)	SEQ	Base space representation of the aligned read. Bases are all uppercase. Equal signs "=" are not used.
c (*category*)	2nd FLAG bit (proper pair) and calculation	The most commonly-used category value, AAA, corresponds to the proper pair bit in the FLAG field. Use POS, MPOS, and strand bits in the FLAG field to calculate other categories. You can find C** pairs (different contigs) by interrogating the mate reference (MRNM). The D** is indicated by the fourth FLAG bit.
g (*color-space read*)	CS attribute	The color read is stored in the CS attribute. However, unlike the 'g' GFF attribute, the color read is stored in an unaltered *.csfasta form that includes the last primer base.
mq (mq [mapping quality])	MAPQ (or SM for pairs)	The mapping quality is stored in the MAPQ field for fragment libraries and in the SM attribute for paired results.
o (*offset*)	CIGAR (hard-clipping)	The offset for an alignment can be computed by checking the number of hard clip (H) operations on the CIGAR string.
p (*mappability ambiguity*)	—	Use mapping quality for uniqueness determinations.
pq (*pairing quality*)	MAPQ (for pairs)	The pairing quality is stored in the MAPQ field for paired results.
q (*color qualities*)	CQ	The CQ attribute stores color quality. The values are not represented as integers, but are represented by ascii-33 encoding.
r (*reference color at mismatches*)	Calculate reference color at mismatches using the samtools `fillmd` command.	CIGAR strings do not distinguish between matches and mismatches. Calculate this information using the samtools `fillmd` command, though in a form that is different from the rb GFF attribute.
s (*color annotations*)	—	—
u (*accumulated mismatch count*)	Per-alignment mismatches supported with CM attribute	The CM attribute specifies the number of color mismatches for the current alignment. You can use a combination of of the number of color mismatches across alignments in the *.bam file to reconstruct values like the 'u' attribute.

Match file format description

The output of mapping steps in BioScope™ Software Resequencing and Whole Transcriptome analyses is a match file, signified by the *.ma file extension. The file is a version of the *.csfasta file generated by primary analysis, enhanced with a comma separated list of hit descriptors.

Match files are not used directly by tertiary analysis tools as of BioScope™ Software v1.2. Third parties that wish to consume match information should not use match files directly, but should develop against the BAM file secondary analysis output.

Before BioScope™ Software v1.2, the mapreads tool and the Mapping tool have produced a match file with additional anchor-extend information (see Figure 107).



```
>102_1361_1882_F3,1_-92804525.2:(43.4.0):q96
T13130313203002203333330222033100210122102131013021
```

Figure 107 Anchor-extend match file fields

In BioScope™ Software v1.2, Whole Transcriptome Analysis can produce match files with additional data designed to guide subsequent analysis steps. The intron descriptor field (n) has two subfields. The first value after the n is the color between the bases of the flanking exons. The second value is the number of reference bases in the intron. The gene id field (g) provides a dot separated list of gene ids associated with the mapping. This provides additional information for junction finding analysis (see Figure 108).

```
>102_1361_1882_F3,1_-92804525.2:(43.4.0):q96:n34.17893:g10.218
T13130313203002203333330222033100210122102131012021
```

Figure 108 Whole transcriptome extensions to the match file description line

CMAP file format description

The *.cmap file provides a mapping between individual fasta reference files, an integer index and a short name and is normally used to describe the chromosomes of an organism. Additional annotation is added in subsequent columns. File paths can be relative or absolute. If relative, the data.dir comment field must be specified. *.fasta reference files must be single entry. See Table 67 on page 308 for a description of *.cmap file parameters. See Table 68 on page 308 for a description of the Human CNV .cmap file parameters.

The CMAP file may begin with one or more comment lines beginning with a pound sign (#). The comments can include file variables. Variable names use dot separated syntax and must appear immediately after the pound sign. Variable values are separated from the name by an equal sign (=) which may be surrounded by one or more white-space characters. The following fields are currently supported:

data.dir

You can use this optional field to define a root directory. If defined, relative file paths used in the data columns are relative to the root directory.

Header

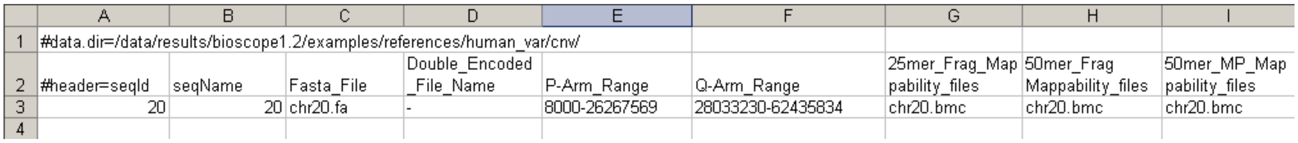
This optional field may be used to define column headers for the columns. Column headers should be tab-separated.

Table 67 CMAP file parameters

Column	Field name	Type	Example	Description
1	Seq id	Non-negative integer	2	The seq id is an ordinal index of the sequence.
2	Seq name	String	2	The name of the sequence. The name is typically a chromosome name and might include X, Y ,M, etc. The *.chr suffix is added to the sequence name in a number of tools to support the UCSC browser function.
3	Fasta file path	String (path)	—	The path to a *.fasta record containing a single sequence entry. The path can be relative or absolute. If the path is relative, you must define the data.dir field in the comments.
4	Double-encoded file path	String (path)	—	If there is a double- encoded version of the *.fasta reference file, the path is specified here. Like the *.fasta file path, if a relative path is provided, data.dir must be specified.
5, 6, 7, etc.	Additional data	Any	—	Any number of additional fields can be specified after the first four columns. These fields can be integers, strings, file paths, etc. If additional fields are file paths, the rules for relative file paths apply.

Table 68 Human CNV *.cmap file description

Field name	Column	Example
P-Arm range	5	50000-50740429
Q-Arm range	6	54450781-134452095
25mer Frag Mappability File	7	Mappability/25.2/chr11.bmc
50mer Frag Mappability File	8	Mappability/50.2/chr11.bmc
50x2 MP Mappability File	9	Mappability/MP_50x2/chr11.bmc



	A	B	C	D	E	F	G	H	I
1	#data.dir=/data/results/bioscope1.2/examples/references/human_var/cnv/								
2	#header=seqId	seqName	Fasta_File	Double_Encoded _File_Name	P-Arm_Range	Q-Arm_Range	25mer_Frag_Map pability_files	50mer_Frag Mappability_files	50mer_MP_Map pability_files
3	20	20	chr20.fa	-	8000-26267569	28033230-62435834	chr20.bmc	chr20.bmc	chr20.bmc
4									

Figure 109 *.cmap file format example

Reference file data overview

Nearly all BioScope™ Software tools use reference sequence data and, in some cases, the tools also use reference annotations and metadata. The *de novo* assembly applications do not use reference data.

Reference data takes a number of forms:

Contig multi-fasta file

A single, multiple-entry *.fasta file where each entry corresponds to a genomic contig or chromosome.

Single contig FASTA file

A single-entry *.fasta file containing one contig or chromosome.

*.CMAP file

A tab-delimited text file. Each line specifies the path to a single contig *.fasta file. Columns in the *.cmap file provide annotation and paths to other chromosome specific support files (see File Formats section).

*.GTF file

A genome annotation file. This file describes gene models used by the Whole Transcriptome pipeline. It is similar in form to the *.gtfs downloaded from the UCSC web site, but includes some changes for WT processing.

Genomic reference files downloaded from UCSC; for example, you can download hg18 from:

http://hgdownload.cse.ucsc.edu/goldenPath/hg18/bigZips/chromFa.zip

The file can be used as a basis for the reference sequence data in BioScope™ Software tools. A few small transformations are required to prepare the *.gtf files for use by BioScope™ Software applications.

Reference sequence data validation

The `reference_validation.pl` script provided with BioScope™ Software checks and corrects common issues with *.fasta sequence files that can cause errors in BioScope™ Software runs. Examples of errors generated by an uncorrected *.gtf file can include extra spaces and mixed case.

Concatenation

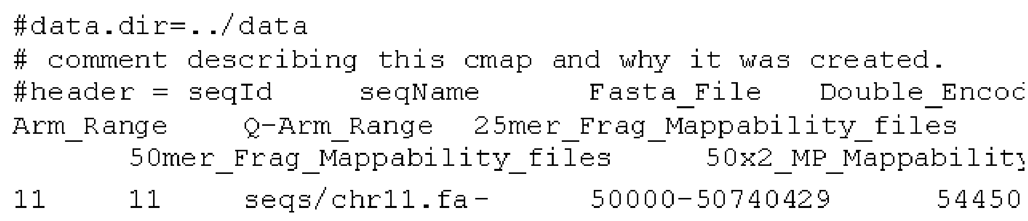
The contig multi-*.fasta reference file can be generated by concatenating the individual chromosome files together.

Annotation

The *.cmap file form is a tab-delimited text file that provides annotation for the contigs associated with a genome.

*.cmap file example

Figure 110 shows an example from a single line of the Human CNV *.cmap file.



```
#data.dir=../data
# comment describing this cmap and why it was created.
#header = seqId     seqName       Fasta_File    Double_Encod
Arm_Range     Q-Arm_Range   25mer_Frag_Mappability_files
       50mer_Frag_Mappability_files      50x2_MP_Mappability

11      11      seqs/chr11.fa-      50000-50740429      54450
```

Figure 110 CMAP example file

Lines one to three are comments, including a line with field headers. Line four is a data line describing one contig (chromosome 11) . The first four columns: (seqId, seqName, Fasta_File, and Double_Encoded_File_Name) are fixed fields. The he first three fields are required. The remaining fields specify the location of mappability files and processing ranges that exclude repeat rich telomeres and centromeres.

Select a reference file

Be sure to use the correct reference file type for your requirements (see Table 69). The identifiers in the *.fasta definition line is carried through many tools, including the *.bam file generated by mapping and pairing, and the *.gff files typically written by tertiary analysis tools.

For some downstream viewing and integration tools, for example, the UCSC genome browser, the identifiers can be used to connect sequence references with other sources of annotation.

Table 69 Reference data required per tool

Tool	Parameter	Reference type
Mapping	Reference	Contig multi FASTA
Pairing	reference	Contig multi FASTA
diBayes	reference	Contig multi FASTA
Small Indel	cmap	CMAP
Human CNV	cmap	CMAP
Large Indel	none*	CMAP
Inversion	non	—

*The Large Indel tool uses the UR field in the *.bam header when the UR field is available and valid. If the UR field is not valid, you must supply a cmap file.

B

Use the SOLiD™ 4 Accuracy Enhancer Tool

This appendix covers:

SOLiD™ Accuracy Enhancer Tool overview

When you use SAET, mapping becomes more accurate, the number of reads with zero miscalls triples, and the number of mapped reads increases by 40 to 50%. The SNP calling on error-corrected data results in up to twice more true positive calls, and a slight increase or decrease in number of false positive calls. Using the De novo assembly on error-corrected reads results in an increase that is as much as three to five times higher in average contiguous length. SAET performance was tested on various datasets. The datasets included a large spectrum of genome sizes and complexities, coverages, and read lengths. SAET shows similar performance on datasets sequenced from enriched regions or genomes of 1 Kbp to 200 Mbp, with coverage from 10 to 4000x and a read length 25-75bp from datasets generated for targeted resequencing, whole transcriptome, and bacterial de novo assembly.

You can run SAET to correct reads sequenced from regions larger than 200 Mbp. However, that scenario requires a high-memory compute node, for example, 48 Gb for 1 Gbp region. SAET can be applied for diploid organisms and for rare variant detection datasets where frequency of rare variant is at least twice that of the color calling error rate. Otherwise, SAET is less effective.

SAET can be used for Whole Transcriptome and Resequencing of bacterial genomes.

SAET implementation

SAET implements a modified version of the spectral alignment error correction algorithm proposed by Pevzner et al., 2001. SAET extends the original algorithm by taking quality values and properties of color-space into account, thus significantly improving the performance of the original method. The method corrects reads given a set of k-mers (seeds) spectrum. Given a set of reads (*R*), and a frequency threshold (*m*), the spectrum is defined as the set of all *k*-mers that appear at least *m* times in reads from *R*. The set of such trusted *k*-mers approximates the set of all *k*-mers in the genome. A read is defined as error-free if all of its *k*-mers are trusted. If all *k*-mers, SATE attempts to make a read error-free by mutating a few colors in the read. SAET only considers substitution mutations since there no insertions or deletions in SOLiD™ system reads. When making error corrections in the read, SAET gives priority to positions with missing calls ".", followed by positions that have lower-quality values. In addition, SAET avoids correcting two adjacent colors with quality value of color-calls above them, thus preserving any SNP evidence. SAET computes the *k*-mer size and the multiplicity of trusted *k*-mers, based on the estimated reference length and number of reads in the dataset. Using default parameters, the color call changes introduced by SAET are expected to be 99.99% accurate. SAET supports multi-core runs. SAET provides a mechanism for multi-node runs by creating sub-spectra from subsets of reads. SAET calculates the possibility of merging the sub-spectra into the final spectrum that can be used for correction of any subset of reads.

SAET input files

By default, SAET takes as input a file with SOLiD™ system reads in *.csfasta format, and a file with quality values in *.qual format (if available), and an expected length of the sequenced region, such as the:

- Genome size for whole genome sequencing.
- Size of enriched region in targeting resequencing, or the
- Size of transcribed region in whole transcriptome sequencing.

The title of each read in the *.csfasta file and the first two characters are irrelevant. The missing colors are encoded as dots. Both the *.csfasta and *.qual files can contain comments and descriptions. SAET constructs a spectrum of all *k*-mers from the set of all reads, and then each read is corrected individually if necessary.

SAET writes the corrected reads and quality values into new *.csfasta and *.qual files, respectively. The new *.qual file uses a zero to replace all quality values corresponding to the corrected color call.

For more information, see "Input files" on page 317.

SAET runtime

The SAET runtime depends on the input size and number of global/local rounds. With a large number of global/local rounds, an expected throughput is 1 Gb per hour. The amount of used RAM is approximately 2 Gb for 1 to 10 Mb references, and less than 6 Gb for 10 to 50 Mb references. Occasionally the amount of RAM used shows higher memory usage that is higher than the actual amount used, because of the memory caching performed by the cluster operating system.

Algorithm/script description

The SAET algorithm has three steps as described below.

Spectrum building

In this step, a temporary file with spectrum constructed from all of the reads is generated in the output directory. This process is very fast and uses no system resources. It creates temporary swap files when memory usage is above 2 Gb.

Error correction

The error correction step takes each read in the input file and attempts to correct it if any of its *k*-mers are not present in the spectrum. This step is the most time-consuming. You can run SAET on multiple cores to improve the error correction speed.

Generating new quality value file

In this step, a new *.qual file is generated from the original *.qual file. In the new *.qual file, the value of each corrected position is zero.

During runtime, SAET outputs a log file containing the log messages generated by the algorithm.

Advanced options

SAET includes new options for developers:

- Spectrum reading and writing
- Trimming or filtering if erroneous reads
- Overwriting spectral frequency
- Support for vote cutoffs
- Support for multi-core running
- Support for running SAET on low-memory machines

Using advanced options

You can use the advanced options available in SAET to perform more or less aggressive correction, or to perform slower but more accurate error correction. Table 70 on page 314 describes the advanced option parameters.

If you trust the quality of your reads more or less than the value specified in the `saet.trustprefix` parameter, then make the corresponding changes to the value. The runtime and the level of error correction aggressiveness depend mainly on the values defined for `saet.localrounds` and `saet.globalrounds`. The `saet.localrounds` parameter allows you to correct up to `saet.localrounds` errors in a read by using a precomputed spectrum. The `saet.globalrounds` parameter recomputes the spectrum after each global round, and allows you to correct up to `saet.localrounds` errors in a read based on the precomputed spectrum.

SAET is designed to reduce the error rate in the reads generated by the SOLiD™ System. Reducing the error rate increases the number of mapped reads for resequencing projects, which can lead to an increase in TP and FP SNP-calls. To decrease FP calls while slightly reducing the number of TP calls, use the updated *.qual file generated in the third step of the SAET algorithm. Use the `saet.qvhigh` parameter to restrict corrections to positions that have a quality value below the `saet.qvhigh` threshold.

Table 70 BioScope™ Software SAET advanced option parameters

Parameter name	Default value	Description
saet.fixdir	"fixed"	The output spectrum and fixed reads are written to the specified directory.
saet.trustprefix=len	0.8 * readLength	Use only the first trustprefix positions of the reads to build the spectrum.
saet.localrounds=lr	readLength/8	Corrects up to localrounds errors in a read. Reduce the value if you observe over-corrections, and increase if under-corrections are observed.
saet.globalrounds	1	Repeat recursively globalrounds times the error correction procedure. Decrease the value if you observe over-corrections and increase if you observe under-corrections.
saet.qvupdate	0	Include a set of 1 if it is necessary to generate a new quality value file.
saet.qvhigh	25	Avoids correction of positions with quality value of qvhigh.
saet.nosampling	0	Avoids random sampling in spectrum building. If the parameter is set to 0, then a subset of reads is used in spectrum building for large datasets (coverage > 300x).
saet.numcores	—	If multi-threading is supported, then increase numcores to run the code in numcores parallel threads.

Using developer options

The developer options available in SAET allow you to modify parameters. For example, you might find that the globally computed cutoff for frequency of trusted seeds does not meet your purpose. For example, the cutoff might be too low and too many junk seeds are considered correct. However, if the cutoff is too high, then many correct but low-frequency seeds are filtered out. You can change the `saet.trustfreq` parameter to overwrite the estimated frequency cutoff. If you noticed that SAET makes many corruptions in the regions of reads with highly packed errors, you can increase `saet.suppvotes` to improve the tendency to correct only isolated errors. SAET provides options for reading and writing spectrum files. The options enable you to build spectrum from better-quality reads and use the spectrum to correct lower-quality reads. You can also building spectrum from a reference, or build a spectrum from multiple files, such as data from multi-run experiments. In certain applications, it is important to trim and filter out error-prone reads. Trimming and filtering is enabled by the `saet.maxtrim` and `saet.trimqv` parameters. Table 71 describes the options available to SAET developers.

Table 71 SAET developer option parameters

Parameter	Default	Description
saet.seed t	optimal	The size of the seed used in spectrum construction.
saet.trustfreq freq	–	Use this option to overwrite estimated frequency cutoff of trusted seeds. All seeds with frequency < "freq" are filtered out of the spectrum.
saet.suppvotes vn	Default vn = 2.	Require at least *vn* separate votes to fix any position. Increase the default value if overcorrection is observed.
saet.outspectxt	–	Outputs spectrum in .txt format in fixed/reads.csfasta.spect.txt. The file includes only seeds with trustable frequencies.
saet.outspecdist	–	Outputs the distribution of frequencies in the spectrum.
saet.outspecbin	–	Outputs spectrum in binary format in fixed/reads.csfasta.spect.bin. The file includes only seeds with trustable frequencies. The file is designed to be loaded later for correction of the reads. If this option is included, then the program stops after generating the spectrum file, and no read correction is performed. You can use his option for parallelization by splitting spectrum generation into multiple jobs, where each job generates a subspectrum from the subset of reads. Outputs spectrum in binary format in the file fixed/reads.csfasta.spect.bin. The file includes seeds with frequency >= 1. If more than two blocks are merged, then frequency is >= freq where -trustfreq freq is provided. The file is designed to be loaded later for correction of the reads. If this option is included, then the program stops after generating the spectrum file, and no read correction is performed. You can use his option for parallelization by splitting spectrum generation into multiple jobs, where each job generates a subspectrum from the subset of reads.
saet.inspecbin files	–	Uses pre-generated file(s) with spectrum in binary format for error correction. Use "," to separate multiple files. All input spectrum files are merged into one spectrum, and a frequency cutoff is applied before correction. All files must have the same seed size. Current reads do not contribute to the spectrum because they are corrected based on input spectra. This option, coupled with the previous option, allows you to use spectrum files generated from higher-quality set of reads, multiple sets of reads, or from reference sequences to correct current reads. You can use the files for parallelization by splitting error correction into multiple jobs, and then correcting a subset of reads.
saet.maxtrim mt	–	Trims erroneous tails of reads up to first trusted seed or up to "mt". If the remaining part of a read is shorter than seed size + 2, then the read is discarded. Do not use this option with the -qvupdate option.
saet.trimqv tq	–	Trims erroneous tails of reads up to the first trusted seed or up to a position with a quality value that is higher than "tq". If the remaining part of a read is shorter than seed size + 2, then the read is discarded. Do not use this parameter with the -qvupdate option.
saet.log filename	–	Outputs the run progress into a log file.

Run SAET examples

Example 1 of the saet.ini parameters

1. Log into the BioScope™ Software cluster.

2. Navigate to the `saet.ini` file.

3. Edit the parameters:

```
saet.run=1
saet.input.csfastafile=${base.dir}/reads1/reads.csfasta
saet.input.qualfile=none
saet.refLength=20000
saet.log=${log.dir}/saet
saet.qvupdate=
```

4. Run the saet program.

A successful run results in the creation of the new directory `/fixed`. The directory contain the corrected reads*.csfasta and the updated *.qual file. During runtime, SAET generates saet.log.txt, a file that contains a summary of the SAET run. The file `saet.log.txt` is also used to output an analysis of the spectrum when developer parameters are used.

Example 2 of saet.ini

1. Login to the BioScope™ Software cluster.

2. Navigate to the `saet.ini` file.

3. Edit the parameters:

```
saet.run=1
saet.input.csfastafile=${base.dir}/reads.csfasta
saet.input.qualfile=${base.dir}/reads.qual
saet.refLength=20000
saet.log=${log.dir}/saet.log.txt
saet.qvupdate=1
saet.fixdir=fixed_dir
saet.trustprefix=22
saet.localrounds=3
saet.globalrounds=2
saet.qvhigh=10
```

4. At a command prompt, enter:

Example of binary spectrum generation

1. Log into the BioScope™ Software cluster.

2. At a command prompt, enter:

```
./saet_mp sample/reads.csfasta sample/reads.qual 20000 -fixdir
fixed_reads -trustprefix 22 -localrounds 3 -globalrounds 2 -
qvhigh 10 -qvupdate -outspecbin
```

Multi-thread example

1. Log into the BioScope™ Software cluster.

2. At a command prompt, enter:

```
cd saet/sample
```

3. At a command prompt, enter:

```
../saet_mp reads.csfasta reads.qual 20000 -fixdir fixed_reads -
trustprefix 22 -localrounds 3 -globalrounds 2 -qvhigh 10 -
qvupdate -numcores 7
```

A successful run results in creation of a new directory named sample/fixed_reads. The new directory contains the corrected reads.csfasta and the updated reads.qual files. SAET performs three local and two global rounds of correction. In most scenarios, SAET does not correct positions that have quality values higher than ten.

Input files

SAET has one required input file and one optional input file (see Table 72).

The required file is a *.csfasta read, which is typically generated by SOLiD™ System. The file must contain reads in color-space that require correction. The missing colors must be encoded as dots. The title of each read and the first two characters are irrelevant. In some cases, the input file header of the file might contain comments and descriptions.

The optional file is a quality value file (*.qual). The *.qual file has quality values for each read in the *.csfasta file. The order of the reads in the *.csfasta file must be the same as the order of the reads in the *.qual file. If a *.qual file is not available when you run the SAET command, enter “none” in the second “Input:” parameter field.

Table 72 SAET input file parameters

Name	Description
reads.csfasta	The *.csfasta file with original reads (in color-space).
reads.qual	The name of the file that contains the quality values (if available). The order of reads in the *.csfasta file must be in the same order as the quality value file. If the file is not available when you run the SAET command, enter “none” in the second “Input:” parameter field.
refLength	The expected length of the assembled sequence, for example: 4,600,000 for E.coli, 4.6 Mb genome, or 30,000,000 for Whole Human Transcriptome.

Sample input file(s)

Input: `sample/reads.csfasta`

```
>1015_1635_189_F3_I1
T03203100300011200123113 30
>1029_1776_965_F3_I1
T0330120031130.22301200030
```

Input: `sample/reads.qual`

```
>1015_1635_189_F3_I1
27 27 27 27 7 27 21 27 27 26 27 27 26 26 27 8 27 21 27 10 25 15
6 27 17
>1029_1776_965_F3_I1
19 27 14 7 27 27 27 24 14 26 22 27 24 4 27 7 27 26 6 26 5 23 26
11 27
```

Output files

Table 73 describes the SAET output files. The next section provides an example of a reads.csfasta file.

Table 73 SAET output file parameters

Parameter	Description
Input parameters	
fixed/reads.csfasta	The *.csfasta file with corrected reads in color-space.
fixed/reads.qual	The quality value file where quality values of corrected positions are replaced with zero, for use with SNP calling.

Sample output file

Output: `sample/fixed_sample/reads.csfasta`

```
>1015_1635_189_F3_I1
T03203100300011220123113330
>1029_1776_965_F3_I1
T03301200311300223012000030
```

Output: `sample/fixed_sample/reads.qual`

```
>1015_1635_189_F3_I1
27 27 27 27 7 27 21 27 27 26 27 27 26 26 27 0 27 21 27 10 25 15
6 27 17
>1029_1776_965_F3_I1
19 27 14 7 27 27 27 24 14 26 22 27 24 0 27 7 27 26 6 26 5 23 26
11 27
```

SAET usage guidelines and parameters

You must login to the BioScope™ Software cluster to run the SAET application.

Usage guidelines

- Run SAET before you run the resequencing or whole transcriptome mapping/pairing tools
- You can use the SAET *.csfasta output with resequencing and WT mapping and pairing.

Usage parameters

To start the SAET application, enter:

```
saet_mp <reads.csfasta> <reads.qual> <refLength> [-options]
```

C

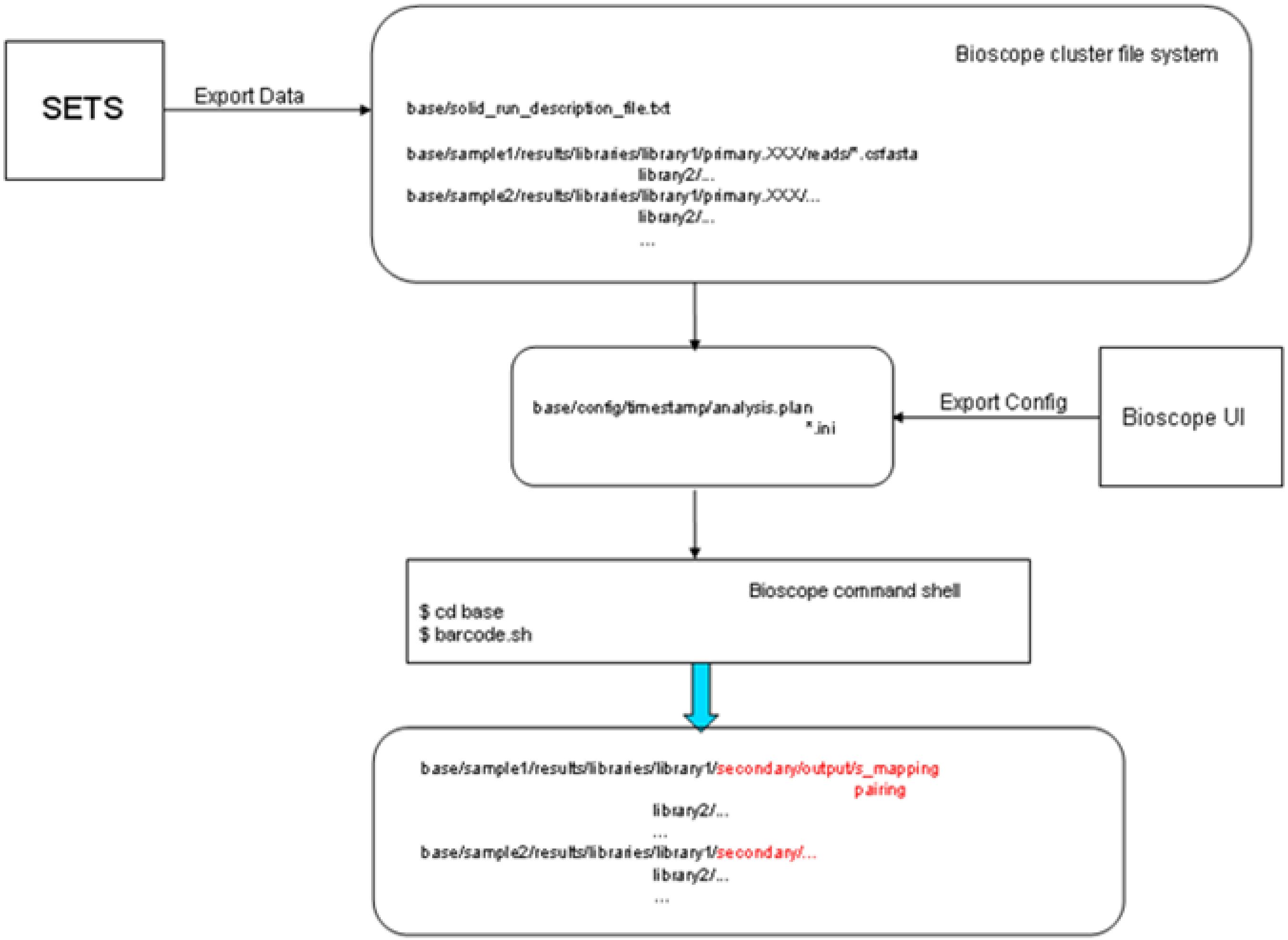
Batch Analysis of Barcoded Library Data

This appendix covers:

Barcode script overview

The barcode script is a program that can run secondary or tertiary tests simultaneously on up to 96 barcoded libraries. Figure 111 provides an overview of the barcode script workflow.

For information about creating barcoded libraries and exporting them from the instrument to BioScope™ Software, see the *Applied Biosystems SOLiD™ 4 System SETS Software User Guide* (4448411). SETS is an application that helps to manage and administer the instrument.


SETS
Export Data
Bioscope cluster file system
base/solid_run_description_file.txt
base/sample1/results/libraries/library1/primary.XXX/reads/*.csfasta
library2/...
base/sample2/results/libraries/library1/primary.XXX/...
library2/...
...
base/config/timestamp/analysis.plan
*.ini
Export Config
Bioscope UI
Bioscope command shell
base/sample1/results/libraries/library1/secondary/output/s_mapping
pairing
library2/...
...
base/sample2/results/libraries/library1/secondary/...
library2/...
...

```
$ cd base
$ barcode.sh
```

Figure 111 Barcode batch file workflow diagram

Prerequisites

You must export the barcoded libraries from the instrument. See Appendix D, “Auto-Export” on page 325. Henceforth, the top level folder of the exported data shall be referred to as "exported base folder".

The exported base folder structure from SETS must be maintained (see Figure 112 on page 321).

Verify that the sample run description file (.txt extension) is present in the exported base folder.

You must know:

- The path to the exported base folder.
- How to login to the BioScope™ Software cluster and run basic UNIX commands
- The location of the plan file that you plan to use.

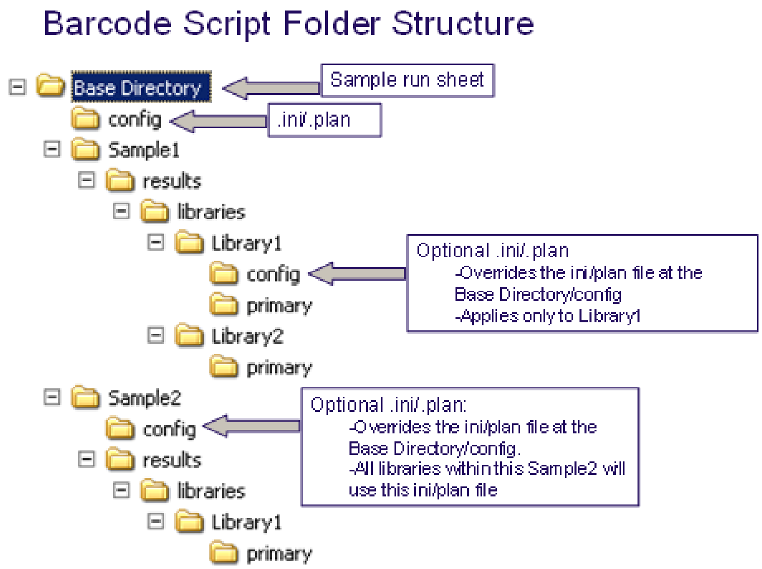


Figure 112 Barcode script folder structure example

Preparing the analysis configuration files

Create an analysis plan from the Bioscope™ Software UI

1. Launch a browser and enter the URL of the BioScope™ Software server.
2. Select a resequencing or WTA tool and set up an analysis.
3. In Global Settings, set the Base Folder path to the exported base folder. For the reads file parameters, choose any *.csfasta file as a placeholder. The barcode script will automatically replace these with the library reads files.
4. Click **ExportConfig**.

After you click ExportConfig, a directory named config shall be created in the top level directory of the barcode data. This directory contains the configuration files that you will need to use.

Creating an analysis plan manually

1. Connect to the BioScope™ Software cluster.

2. Copy the *.ini and *.plan files from the barcode directory in the BioScope™ Software examples to a convenient location, preferably the exported base folder.

3. Modify the plan and ini files as desired from the example template. Leave the reads files parameters empty (the barcode batch script shall automatically replace these with correct values).

Running the barcode script

The barcode.sh script takes a specified plan file and executes it on the library data in the current working folder in batch mode. The list of libraries on which to run the analysis is read from the run description file.

By default, each library is analyzed in serial (though for a single library, the analysis is parallelized as specified in the plan file). An option is provided to execute the analysis jobs for all libraries in parallel. This mode is not recommended in general, as it could create several queued cluster jobs and may overload the cluster. Use this option only in cases where the cluster could handle analyzing several libraries at once.

How to run the barcode script

1. Create an analysis configuration (plan + ini files) per the instructions in the above section.

2. Login to the BioScope™ Software server.

3. Change to the exported base folder of the barcode data. The script must be run from this folder, otherwise it will fail.

4. Enter:

```
barcode.sh [path to plan file]
```

Usage parameters

Table 74 describes the usage parameters of the `barcode.sh` script.

Table 74 barcode.sh usage parameter descriptions

Parameter	Description
-o [path to output directory]	The output directory. By default, the script creates the output in a sub-directory called output in the current folder.
-p	Execute analysis jobs for all libraries in parallel. Not recommended in general.
-d	Dry run. Create output directory structure and per-library BioScope™ Software plan files, but do not run the actual analysis
-r [path to run description file]	The run description file to use. By default, the script uses the first .txt file it finds in the current directory.

Advanced usage

How to modify the list of libraries to analyze

The script picks up the list of libraries from the run description file present in the top level directory of the barcoded data. By default, this includes the entire list of libraries from the experiment. To remove certain libraries from being analyzed, comment out or delete the lines corresponding to those libraries (to comment out a specific line, prefix the line with a "#" character).

How to use different configuration files for different libraries

It might be the case that you may need to use different reference files or other parameters for specific libraries (for example, say, for the last 4 out of 20 libraries in a run). In such cases, there are two options.

Option 1

Run the barcode script multiple times for each set of libraries. In the above example, you would create a configuration and run it for 16 libraries first (by commenting out the last 4 libraries from the run description file). Then you would modify the configuration as desired (by changing the *.ini files) and run the script on the remaining 4 libraries (now by commenting out the first 16 libraries in the run description file).

Option 2

Use the script's plan file overriding mechanism. The script supports a way to override the default plan file provided in the command line, with a per-library or per-sample plan file. To do this, copy the overriding plan file to the four library folders (sampleX/results/libraries/library) that need to be analyzed differently. The script shall use the most specific configuration provided for each library.

D

Auto-Export

This appendix covers:

Export overview

You have two options when deciding how to export primary test results from the instrument to the BioScope™ Software cluster.

- Use rsync or a similar program to manually copy the files from the instrument to the BioScope™ Software cluster. For more information, contact your system administrator.
- You can use the auto-export feature, which automatically copies the results of the primary test results to the BioScope™ Software cluster.

Note: Auto-export is available only if the BioScope™ Software installer selected the full-install option.

The rest of this appendix describes how to use the auto-export feature.

When you auto-export datasets from the instrument to BioScope™ Software, the SETS server on the instrument establishes a network connection to the Linux server where BioScope™ Software is installed and copies the datasets to the BioScope™ Software directory.

BioScope™ Software stores the exported data in a directory structure that is identical to the directory structure of the dataset on the instrument. The auto-export feature is compatible with barcoded and non-barcoded libraries.

Tip: For information about using a batch file to run tools on exported barcoded libraries, see Appendix B, "Use the SOLiD™ 4 Accuracy Enhancer Tool" on page 311.

Figure 113 on page 327 identifies the auto-export components. Figure 114 on page 328 shows an example of the structure created for a barcoded library exported to BioScope™ Software.

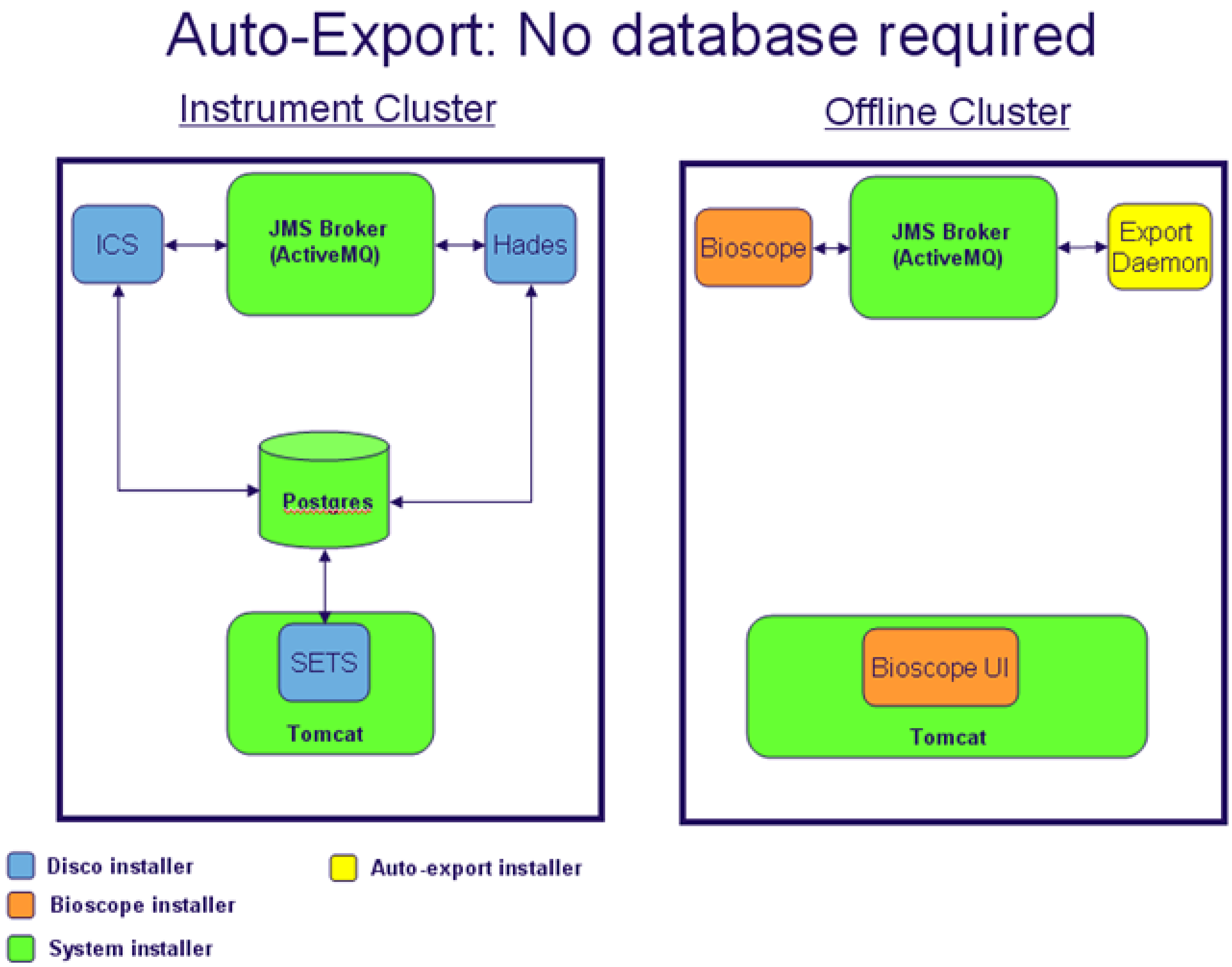


Figure 113 Auto-export components

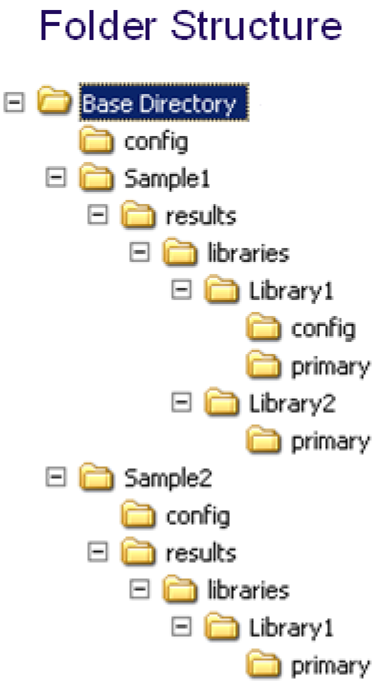


Figure 114 Exported folder structure example (barcoded library)

Configuring auto-export on BioScope™ Software

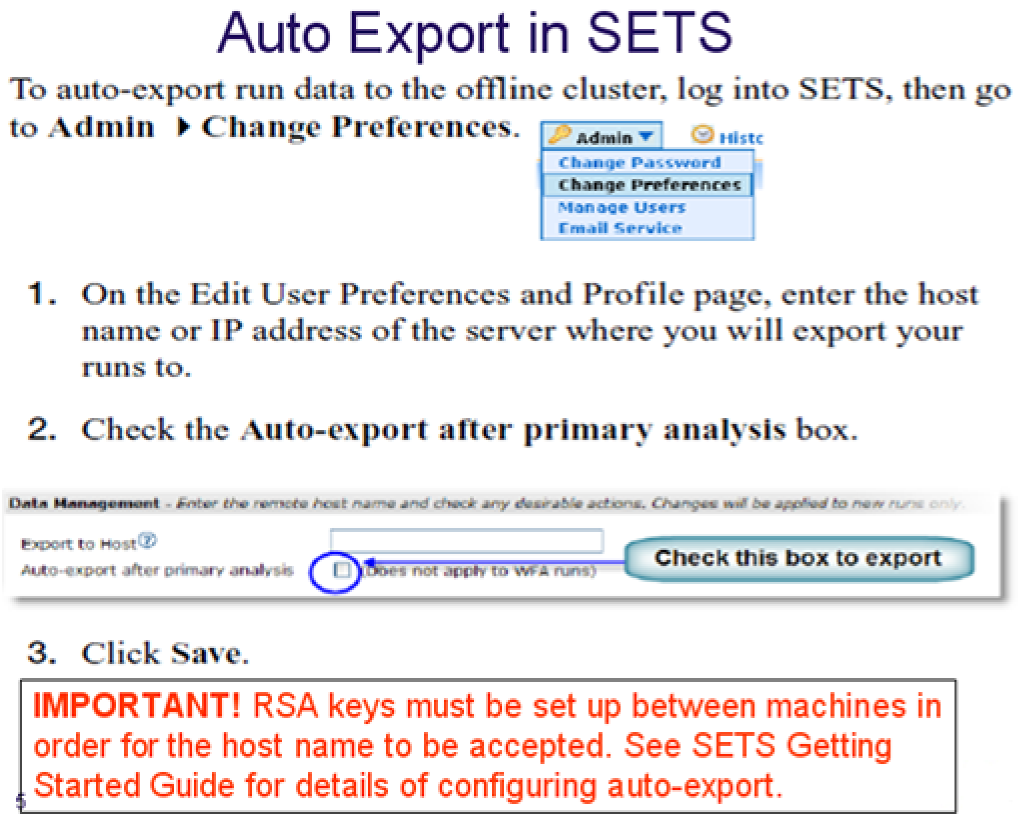
Configuring auto-export on the instrument

1. Login to the instrument.

2. Configure the RSA keys. RSA keys help ensure a secure connection between the instrument and the BioScope™ Software cluster. For information about setting up RSA keys, see the instructions in *Applied Biosystems SOLiD™ 4 System SETS Software User Guide* (4448411).

3. Login to SETS with user-level privileges.

4. Enable auto export in the Preferences menu (see Figure 115). For information about enabling auto export, see the instructions in *Applied Biosystems SOLiD™ 4 System SETS Software User Guide* (4448411).



Auto Export in SETS

To auto-export run data to the offline cluster, log into SETS, then go to **Admin ▸ Change Preferences**.

1. On the Edit User Preferences and Profile page, enter the host name or IP address of the server where you will export your runs to.
2. Check the **Auto-export after primary analysis** box.

Data Management - *Enter the remote host name and check any desirable actions. Changes will be applied to new runs only.*

Export to Host

Auto-export after primary analysis □ (Does not apply to WFA runs)

3. Click **Save**.

IMPORTANT! RSA keys must be set up between machines in order for the host name to be accepted. See SETS Getting Started Guide for details of configuring auto-export.

Figure 115 Auto-export configuration in SETS

Configuring auto-export on BioScope™ Software

1. Log in to the BioScope™ Software cluster as bioscope.
2. Run this command to start the export daemon:

```
solid_java_app.sh com.apldbio.aga.hades.jms.AutoExportDaemon
```

Login to SETS to monitor the progress of the files that are being exported to the BioScope™ Software cluster.

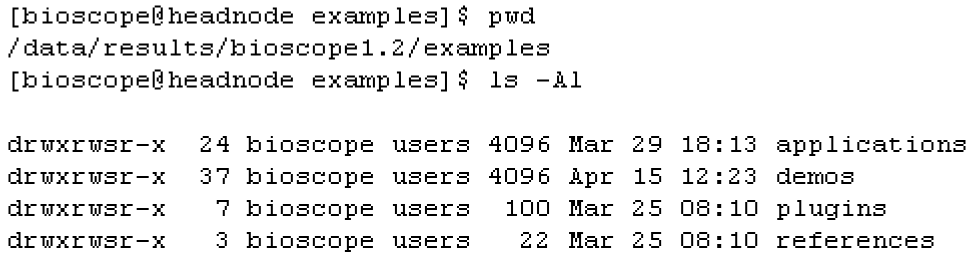
E

Examples

This appendix covers:

Introduction

The examples directory (see Figure 116) contains sample data that you can use to run a tool, view a sample report, modify a *.ini file, and more. The examples directory contains all of the files required to run all sample data.



```
[bioscope@headnode examples]$ pwd
/data/results/bioscope1.2/examples
[bioscope@headnode examples]$ ls -Al

drwxrwsr-x  24 bioscope users 4096 Mar 29 18:13 applications
drwxrwsr-x  37 bioscope users 4096 Apr 15 12:23 demos
drwxrwsr-x   7 bioscope users  100 Mar 25 08:10 plugins
drwxrwsr-x   3 bioscope users   22 Mar 25 08:10 references
```

Figure 116 Examples directory

Install the examples directory

1. Login to **solidsoftwaretools.com/gf/project/bioscope** If you do not have an account, contact your Lifetech account representative.

2. Download `BioScope-1.2.1.examples.tar.gz` to the head node of the Linux cluster where you plan to install, or have installed, BioScope™ Software.

3. Copy `BioScope-1.2.1.examples.tar.gz` to `/data/results/bioscope1.2/examples`.

4. Create an account called "bioscope".

5. Add user "bioscope" to the users group.

6. At a command prompt, enter
   `tar -xvzf BioScope-1.2.1.examples.tar.gz`
   to untar the examples image.

Before you begin

Decide if you want to use the command line or the web interface to work with the sample data in the examples directory.

Read the README files in the demos and applications directories for the latest information about prerequisites and system requirements.

Demos README file

1. Log in as "bioscope" to the BioScope™ Software head node.

2. At a command prompt, enter:

   ```
   $cd /data/results/bioscope1.2/examples/demos
   ```

3. Open the README file in a text editor.

4. Complete all prerequisites.

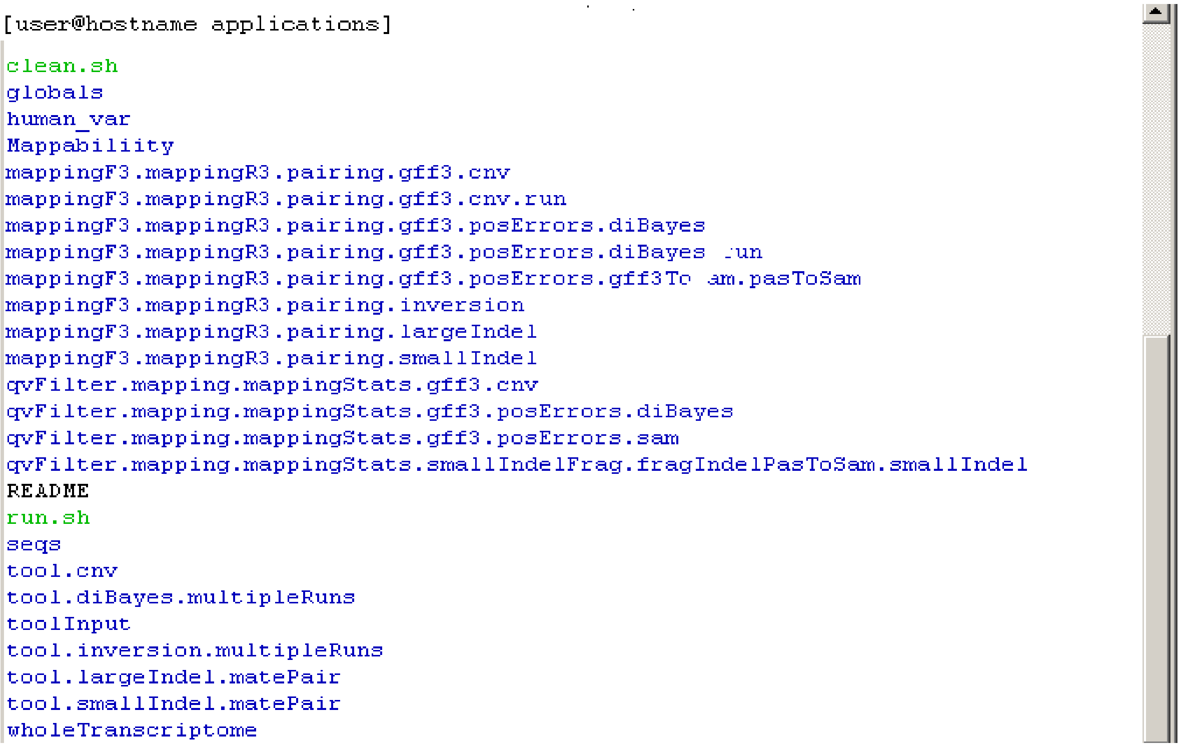
Applications README file

1. Log in as “bioscope” to the BioScope™ Software head node.
2. At a command prompt, enter:
   ```
   $cd /data/results/bioscope1.2/examples/applications
   ```
3. Open the README file in a text editor.
4. Complete all prerequisites.

Applications overview

The applications directory (see Figure 117) contains sample programs that use human chromosome 11, 12 and 20 data as input. You can only run the sample applications via the command line.

IMPORTANT! The mapping sample writes temporary data to the input data folder when you run an application. To prevent collision across runs, do not run multiple application examples at the same time.



```
[user@hostname applications]
clean.sh
globals
human_var
Mappability
mappingF3.mappingR3.pairing.gff3.cnv
mappingF3.mappingR3.pairing.gff3.cnv.run
mappingF3.mappingR3.pairing.gff3.posErrors.diBayes
mappingF3.mappingR3.pairing.gff3.posErrors.diBayes.run
mappingF3.mappingR3.pairing.gff3.posErrors.gff3ToSam.pasToSam
mappingF3.mappingR3.pairing.inversion
mappingF3.mappingR3.pairing.largeIndel
mappingF3.mappingR3.pairing.smallIndel
qvFilter.mapping.mappingStats.gff3.cnv
qvFilter.mapping.mappingStats.gff3.posErrors.diBayes
qvFilter.mapping.mappingStats.gff3.posErrors.sam
qvFilter.mapping.mappingStats.smallIndelFrag.fragIndelPasToSam.smallIndel
README
run.sh
seqs
tool.cnv
tool.diBayes.multipleRuns
toolInput
tool.inversion.multipleRuns
tool.largeIndel.matePair
tool.smallIndel.matePair
wholeTranscriptome
```

Figure 117 Examples applications directory

Demos overview

The demos directory (see Figure 118 on page 335) contains sample programs that use ecoli DH10B as input data. You can run all demos from the command line. You can run a subset of demos from the web interface.

Run a demo from the command line

1. Login as "bioscope" to the BioScope™ Software head node.
2. At a command prompt, enter:
   ```
   $cd /data/results/bioscope1.2/examples/demos
   ```
3. Select the demo that you want to run.
4. At a command prompt, enter:
   ```
   $cd /<demo_name>
   ```
5. At a command prompt, enter:
   ```
   run.sh
   ```
6. To view the results of the run, open the *.ini file associated with the `<demo_name>`.
7. Go to the log directory specified in the *.ini file to view the results.

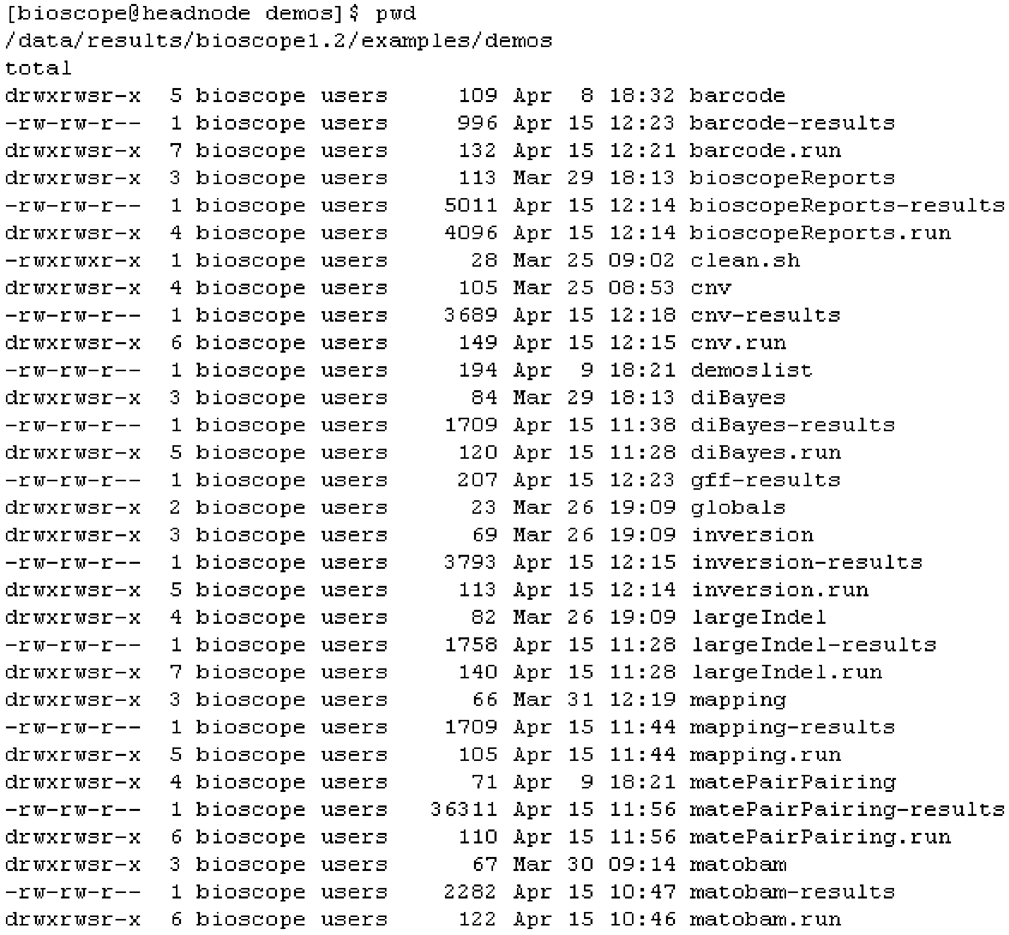
Run a demo from the web interface

You can run the following demos from the BioScope™ Software GUI:

- Map Data
  - Follow the instructions in "Run the Map Data tool from the web interface" on page 127.
- Find Human CNVs
  - Follow the instructions in "Run the Find Human CNVs tool from the web interface" on page 209.
- Large Indel
  - Follow the instructions in "Run the Find Large InDels tool from the web interface" on page 253.
- Find SNPs
  - Follow the instructions in "Run the Find SNPs tool from the web interface" on page 187
- Inversion
  - Follow the instructions in "Run the Find Inversions tool from the web interface" on page 233

When you use the web interface to run a demo, be sure that you point to the files in /data/results/bioscope_1.2/examples/

For example, when you run the Find Human CNV tool in the web interface, you are required to enter the path to the *.cmap file. You would enter `/data/results/bioscope1.2/examples/references/human_var/<cmap_file>`.



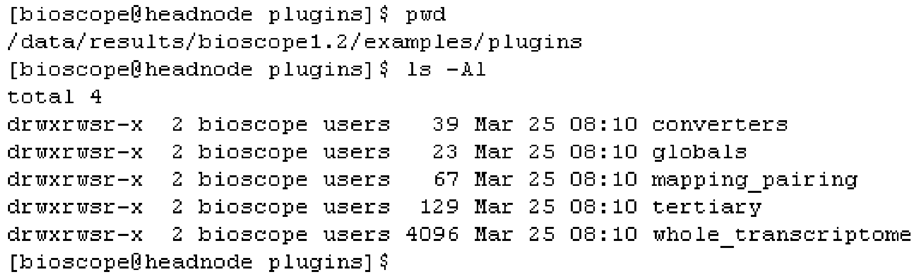
```
[bioscope@headnode demos]$ pwd
/data/results/bioscope1.2/examples/demos
total
drwxrwsr-x  5 bioscope users    109 Apr  8 18:32 barcode
-rw-rw-r--  1 bioscope users    996 Apr 15 12:23 barcode-results
drwxrwsr-x  7 bioscope users    132 Apr 15 12:21 barcode.run
drwxrwsr-x  3 bioscope users    113 Mar 29 18:13 bioscopeReports
-rw-rw-r--  1 bioscope users   5011 Apr 15 12:14 bioscopeReports-results
drwxrwsr-x  4 bioscope users   4096 Apr 15 12:14 bioscopeReports.run
-rwxrwxr-x  1 bioscope users     28 Mar 25 09:02 clean.sh
drwxrwsr-x  4 bioscope users    105 Mar 25 08:53 cnv
-rw-rw-r--  1 bioscope users   3689 Apr 15 12:18 cnv-results
drwxrwsr-x  6 bioscope users    149 Apr 15 12:15 cnv.run
-rw-rw-r--  1 bioscope users    194 Apr  9 18:21 demoslist
drwxrwsr-x  3 bioscope users     84 Mar 29 18:13 diBayes
-rw-rw-r--  1 bioscope users   1709 Apr 15 11:38 diBayes-results
drwxrwsr-x  5 bioscope users    120 Apr 15 11:28 diBayes.run
-rw-rw-r--  1 bioscope users    207 Apr 15 12:23 gff-results
drwxrwsr-x  2 bioscope users     23 Mar 26 19:09 globals
drwxrwsr-x  3 bioscope users     69 Mar 26 19:09 inversion
-rw-rw-r--  1 bioscope users   3793 Apr 15 12:15 inversion-results
drwxrwsr-x  5 bioscope users    113 Apr 15 12:14 inversion.run
drwxrwsr-x  4 bioscope users     82 Mar 26 19:09 largeIndel
-rw-rw-r--  1 bioscope users   1758 Apr 15 11:28 largeIndel-results
drwxrwsr-x  7 bioscope users    140 Apr 15 11:28 largeIndel.run
drwxrwsr-x  3 bioscope users     66 Mar 31 12:19 mapping
-rw-rw-r--  1 bioscope users   1709 Apr 15 11:44 mapping-results
drwxrwsr-x  5 bioscope users    105 Apr 15 11:44 mapping.run
drwxrwsr-x  4 bioscope users     71 Apr  9 18:21 matePairPairing
-rw-rw-r--  1 bioscope users  36311 Apr 15 11:56 matePairPairing-results
drwxrwsr-x  6 bioscope users    110 Apr 15 11:56 matePairPairing.run
drwxrwsr-x  3 bioscope users     67 Mar 30 09:14 matobam
-rw-rw-r--  1 bioscope users   2282 Apr 15 10:47 matobam-results
drwxrwsr-x  6 bioscope users    122 Apr 15 10:46 matobam.run
```

Figure 118 Examples demos directory

Plugins overview

The plugins directory (see Figure 119 on page 336) contains examples of the *.ini files for each BioScope™ Software tool. The *.ini files in the plugins directory contain generic information that is appropriate for running sample data. In a working BioScope™ Software system, the *.ini files contain site-specific information required to run each tool.

After you install BioScope™ Software, you can copy the *.ini files from the plugins directory to your working directory and then customize files with data specific to your working BioScope™ Software system. See "Create the directory structure" on page 37 for more information.

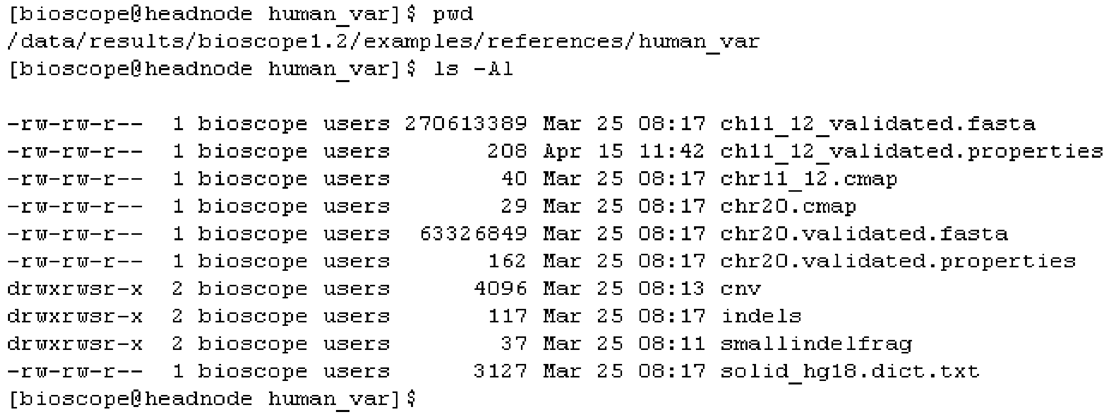


```
[bioscope@headnode plugins]$ pwd
/data/results/bioscope1.2/examples/plugins
[bioscope@headnode plugins]$ ls -Al
total 4
drwxrwsr-x  2 bioscope users   39 Mar 25 08:10 converters
drwxrwsr-x  2 bioscope users   23 Mar 25 08:10 globals
drwxrwsr-x  2 bioscope users   67 Mar 25 08:10 mapping_pairing
drwxrwsr-x  2 bioscope users  129 Mar 25 08:10 tertiary
drwxrwsr-x  2 bioscope users 4096 Mar 25 08:10 whole_transcriptome
[bioscope@headnode plugins]$
```

Figure 119 Examples plugins directory

References overview

The references directory contains the *.fasta, cmap, *.properties and related files required to run the programs in the demos and applications directories (see Figure 120).



```
[bioscope@headnode human_var]$ pwd
/data/results/bioscope1.2/examples/references/human_var
[bioscope@headnode human_var]$ ls -Al

-rw-rw-r--  1 bioscope users 270613389 Mar 25 08:17 ch11_12_validated.fasta
-rw-rw-r--  1 bioscope users       208 Apr 15 11:42 ch11_12_validated.properties
-rw-rw-r--  1 bioscope users        40 Mar 25 08:17 chr11_12.cmap
-rw-rw-r--  1 bioscope users        29 Mar 25 08:17 chr20.cmap
-rw-rw-r--  1 bioscope users  63326849 Mar 25 08:17 chr20.validated.fasta
-rw-rw-r--  1 bioscope users       162 Mar 25 08:17 chr20.validated.properties
drwxrwsr-x  2 bioscope users      4096 Mar 25 08:13 cnv
drwxrwsr-x  2 bioscope users       117 Mar 25 08:17 indels
drwxrwsr-x  2 bioscope users        37 Mar 25 08:11 smallindelfrag
-rw-rw-r--  1 bioscope users      3127 Mar 25 08:17 solid_hg18.dict.txt
[bioscope@headnode human_var]$
```

Figure 120 Examples reference directory

F Software License Agreement

APPLIED BIOSYSTEMS END USER SOFTWARE LICENSE AGREEMENT

FOR INSTRUMENT OPERATING AND ASSOCIATED BUNDLED SOFTWARE
AND LIMITED PRODUCT WARRANTY

Applied Biosystems SOLiD™ 4 System - BioScope™ Software v1.2.1

NOTICE TO USER: PLEASE READ THIS DOCUMENT CAREFULLY. THIS IS THE CONTRACT BETWEEN YOU AND LIFE TECHNOLOGIES REGARDING THE OPERATING SOFTWARE FOR YOUR APPLIED BIOSYSTEMS WORKSTATION OR OTHER INSTRUMENT AND BUNDLED SOFWARE INSTALLED WITH YOUR OPERATING SOFTWARE. THIS AGREEMENT CONTAINS WARRANTY AND LIABILITY DISCLAIMERS AND LIMITATIONS. YOUR INSTALLATION AND USE OF THE APPLIED BIOSYSTEMS SOFTWARE IS SUBJECT TO THE TERMS AND CONDITIONS CONTAINED IN THIS END USER SOFTWARE LICENSE AGREEMENT.

IF YOU DO NOT AGREE TO THE TERMS AND CONDITIONS OF THIS LICENSE, YOU SHOULD PROMPTLY RETURN THIS SOFTWARE, TOGETHER WITH ALL PACKAGING, TO APPLIED BIOSYSTEMS AND YOUR PURCHASE PRICE WILL BE REFUNDED.

This Applied Biosystems End User License Agreement accompanies an Applied Biosystems software product ("Software") and related explanatory materials ("Documentation"). The term "Software" also includes any upgrades, modified versions, updates, additions and copies of the Software licensed to you by Applied Biosystems. The term "Applied Biosystems," as used in this License, means Applied Biosystems, LLC. The term "License" or "Agreement" means this End User Software License Agreement. The term "you" or "Licensee" means the purchaser of this license to use the Software.

THIRD PARTY PRODUCTS

This Software uses third-party software components from several sources. Portions of these software components are copyrighted and licensed by their respective owners. Various components require distribution of source code or if a URL is used to point the end-user to a source-code repository, and the source code is not available at such site, the distributor must, for a time determined by the license, offer to provide the source code. In such cases, please contact your Life Technologies representative. As well, various licenses require that the end-user receive a copy of the license. Such licenses may be found on the distribution media in a folder called "Licenses." In order to use this Software, the end-user must abide by the terms and conditions of these third-party licenses. After installation, the licenses may also be found in a folder named "Licenses" located in the Software installation's root directory.

TITLE

Title, ownership rights and intellectual property rights in and to the Software and Documentation shall at all times remain with Applied Biosystems, LLC and its subsidiaries, and their suppliers. All rights not specifically granted by this License, including Federal and international copyrights, are reserved by Life Technologies or their respective owners.

COPYRIGHT

The Software, including its structure, organization, code, user interface and associated Documentation, is a proprietary product of Life Technologies or its suppliers, and is protected by international laws of copyright. The law provides for civil and criminal penalties for anyone in violation of the laws of copyright.

LICENSE

Use of the Software

1. Subject to the terms and conditions of this Agreement, Applied Biosystems, LLC grants the purchaser of this product a non-exclusive license only to install and use the Software to operate the single product in connection with which this License was purchased and to display, analyze and otherwise manipulate data generated by the use of such product. There is no limit to the number of computers on which you may install and use the Software to display, analyze and otherwise manipulate such data.

2. If the Software uses registration codes, access to the number of licensed copies of Software is controlled by a registration code. For example, if you have a registration code that enables you to use five copies of Software simultaneously, you cannot install the Software on more than five separate computers.

3. You may make one copy of the Software in machine-readable form solely for backup or archival purposes. You must reproduce on any such copy all copyright notices and any other proprietary legends found on the original. You may not make any other copies of the Software except as permitted under Section 1 above.

Restrictions

1. You agree that you will not copy, transfer, rent, modify, use or merge the Software, or the associated documentation, in whole or in part, except as expressly permitted in this Agreement.

2. You agree that you will not reverse assemble, decompile, or otherwise reverse engineer the Software.

3. You agree that you will not remove any proprietary, copyright, trade secret or warning legend from the Software or any Documentation.

4. You agree to fully comply with all export laws and restrictions and regulations of the United States or applicable foreign agencies or authorities. You agree that you will not export or reexport, directly or indirectly, the Software into any country prohibited by the United States Export Administration Act and the regulations thereunder or other applicable United States law.

5. You agree that you will not modify, sell, rent, transfer (except temporarily in the event of a computer malfunction), resell for profit, or distribute this license or the Software, or create derivative works based on the Software, or any part thereof or any interest therein. Notwithstanding the foregoing, if this Software is instrument operating software, you may transfer this Software to a purchaser of the specific instrument in or for which this Software is installed in connection with any sale of such instrument, provided that the transferee agrees to be bound by and to comply with the provisions of this Agreement.

Trial

If this license is granted on a trial basis, you are hereby notified that license management software may be included to automatically cause the Software to cease functioning at the end of the trial period.

Termination

You may terminate this Agreement by discontinuing use of the Software, removing all copies from your computers and storage media, and returning the Software and Documentation, and all copies thereof, to Life Technologies. Life Technologies may terminate this Agreement if you fail to comply with all of its terms, in which case you agree to discontinue using the Software, remove all copies from your computers and storage media, and return the Software and Documentation, and all copies thereof, to Life Technologies.

U.S. Government End Users

The Software is a "commercial item," as that term is defined in 48 C.F.R. 2.101 (Oct. 1995), consisting of "commercial computer software" and "commercial computer software documentation," as such terms are used in 48 C.F.R. 12.212 (Sept. 1995). Consistent with 48 C.F.R. 12.212 and 48 C.F.R. 227.7202-1 through 227.7202-4 (June 1995), all U.S. Government End Users acquire the Software with only those rights set forth herein.

European Community End Users

If this Software is used within a country of the European Community, nothing in this Agreement shall be construed as restricting any rights available under the European Community Software Directive, O.J. Eur. Comm. (No. L. 122) 42 (1991).

Regulated Uses

You acknowledges that the Software has not been cleared, approved, registered or otherwise qualified (collectively, "Approval") by Applied Biosystems, LLC with any regulatory agency for use in diagnostic or therapeutic procedures, or for any other use requiring compliance with any federal or state law regulating diagnostic or therapeutic products, blood products, medical devices or any similar product (hereafter collectively referred to as "federal or state drug laws"). The Software may not be used for any purpose that would require any such Approval unless proper Approval is obtained. You agree that if you elect to use the Software for a purpose that would subject you or the Software to the jurisdiction of any federal or state drug laws, you will be solely responsible for obtaining any required Approvals and otherwise ensuring that your use of the Software complies with such laws.

LIMITED WARRANTY and LIMITATION OF REMEDIES

Limited Warranty. Applied Biosystems warrants that, during the same period as of the SOLiD Analyzer for which this Software is an instrument operating software, the Software will function substantially in accordance with the functions and features described in the Documentation delivered with the Software when properly installed, and that for a period of ninety days from the beginning of the applicable warranty period (as described below) the tapes, CDs, diskettes or other media bearing the Software will be free of defects in materials and workmanship under normal use.

The above warranties do not apply to defects resulting from misuse, neglect, or accident, including without limitation: operation outside of the environmental or use specifications, or not in conformance with the instructions for any instrument system, software, or accessories; improper or inadequate maintenance by the user; installation of software or interfacing, or use in combination with software or products not supplied or authorized by Applied Biosystems; intrusive activity, including without limitation computer viruses, hackers or other unauthorized interactions with instrument or software that detrimentally affects normal operations;.and modification or repair of the products not authorized by Applied Biosystems.

Warranty Period Commencement Date. The applicable warranty period for software begins on the earlier of the date of installation or three (3) months from the date of shipment for software installed by Applied Biosystems' personnel. For software installed by the purchaser or anyone other than Applied Biosystems, the warranty period begins on the date the software is delivered to you. The applicable warranty period for media begins on the date the media is delivered to the purchaser.

APPLIED BIOSYSTEMS MAKES NO OTHER WARRANTIES OF ANY KIND WHATSOEVER, EXPRESS OR IMPLIED, WITH RESPECT TO THE SOFTWARE OR DOCUMENTATION, INCLUDING BUT NOT LIMITED TO WARRANTIES OF FITNESS FOR A PARTICULAR PURPOSE OR MERCHANTABILITY OR THAT THE SOFTWARE OR DOCUMENTATION IS NON-INFRINGING. ALL OTHER WARRANTIES ARE EXPRESSLY DISCLAIMED. WITHOUT LIMITING THE GENERALITY OF THE FOREGOING, APPLIED BIOSYSTEMS MAKES NO WARRANTIES THAT THE SOFTWARE WILL MEET YOUR REQUIREMENTS, THAT OPERATION OF THE LICENSED SOFTWARE WILL BE UNINTERRUPTED OR ERROR FREE OR WILL CONFORM EXACTLY TO THE DOCUMENTATION, OR THAT APPLIED BIOSYSTEMS WILL CORRECT ALL PROGRAM ERRORS. APPLIED BIOSYSTEMS' SOLE LIABILITY AND RESPONSIBILITY FOR BREACH OF WARRANTY RELATING TO THE SOFTWARE OR DOCUMENTATION SHALL BE LIMITED, AT APPLIED BIOSYSTEMS' SOLE OPTION, TO (1) CORRECTION OF ANY ERROR IDENTIFIED TO APPLIED BIOSYSTEMS IN A WRITING FROM YOU IN A SUBSEQUENT RELEASE OF THE SOFTWARE, WHICH SHALL BE SUPPLIED TO YOU FREE OF CHARGE, (2) ACCEPTING A RETURN OF THE PRODUCT, AND REFUNDING THE PURCHASE PRICE UPON RETURN OF THE PRODUCT AND REMOVAL OF ALL COPIES OF THE SOFTWARE FROM YOUR COMPUTERS AND STORAGE DEVICES, (3) REPLACEMENT OF THE DEFECTIVE SOFTWARE WITH A FUNCTIONALLY EQUIVALENT PROGRAM AT NO CHARGE TO YOU, OR (4) PROVIDING A REASONABLE WORK AROUND WITHIN A REASONABLE TIME. APPLIED BIOSYSTEMS SOLE LIABILITY AND RESPONSIBILITY UNDER THIS AGREEMENT FOR BREACH OF WARRANTY RELATING TO MEDIA IS THE REPLACEMENT OF DEFECTIVE MEDIA RETURNED WITHIN 90 DAYS OF THE DELIVERY DATE. THESE ARE YOUR SOLE AND EXCLUSIVE REMEDIES FOR ANY BREACH OF WARRANTY. WARRANTY CLAIMS MUST BE MADE WITHIN THE APPLICABLE WARRANTY PERIOD.

LIMITATION OF LIABILITY

IN NO EVENT SHALL APPLIED BIOSYSTEMS OR ITS SUPPLIERS BE RESPONSIBLE OR LIABLE, WHETHER IN CONTRACT, TORT, WARRANTY OR UNDER ANY STATUTE (INCLUDING WITHOUT LIMITATION ANY TRADE PRACTICE, UNFAIR COMPETITION OR OTHER STATUTE OF SIMILAR IMPORT) OR ON ANY OTHER BASIS FOR SPECIAL, INDIRECT, INCIDENTAL, MULTIPLE, PUNITIVE, OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE POSSESSION OR USE OF, OR THE INABILITY TO USE, THE SOFTWARE OR DOCUMENTATION, EVEN IF APPLIED BIOSYSTEMS IS ADVISED IN ADVANCE OF THE POSSIBILITY OF SUCH DAMAGES, INCLUDING WITHOUT LIMITATION DAMAGES ARISING FROM OR RELATED TO LOSS OF USE, LOSS OF DATA, DOWNTIME, OR FOR LOSS OF REVENUE, PROFITS, GOODWILL OR BUSINESS OR OTHER FINANCIAL LOSS. IN ANY CASE, THE ENTIRE LIABILITY OF APPLIED BIOSYSTEMS' AND ITS SUPPLIERS UNDER THIS LICENSE, OR ARISING OUT OF THE USE OF THE SOFTWARE, SHALL NOT EXCEED IN THE AGGREGATE THE PURCHASE PRICE OF THE PRODUCT.

SOME STATES, COUNTRIES OR JURISDICTIONS LIMIT THE SCOPE OF OR PRECLUDE LIMITATIONS OR EXCLUSION OF REMEDIES OR DAMAGES, OR OF LIABILITY, SUCH AS LIABILITY FOR GROSS NEGLIGENCE OR WILLFUL MISCONDUCT, AS OR TO THE EXTENT SET FORTH ABOVE, OR DO NOT ALLOW IMPLIED WARRANTIES TO BE EXCLUDED. IN SUCH STATES, COUNTRIES OR JURISDICTIONS, THE LIMITATION OR EXCLUSION OF WARRANTIES, REMEDIES, DAMAGES OR LIABILITY SET FORTH ABOVE MAY NOT APPLY TO YOU. HOWEVER, ALTHOUGH THEY SHALL NOT APPLY TO THE EXTENT PROHIBITED BY LAW, THEY SHALL APPLY TO THE FULLEST EXTENT PERMITTED BY LAW. YOU MAY ALSO HAVE OTHER RIGHTS THAT VARY BY STATE, COUNTRY OR OTHER JURISDICTION.

GENERAL

This Agreement shall be governed by laws of the State of California, exclusive of its conflict of laws provisions. This Agreement shall not be governed by the United Nations Convention on Contracts for the International Sale of Goods. This Agreement contains the complete agreement between the parties with respect to the subject matter hereof, and supersedes all prior or contemporaneous agreements or understandings, whether oral or written. If any provision of this Agreement is held by a court of competent jurisdiction to be contrary to law, that provision will be enforced to the maximum extent permissible, and the remaining provisions of this Agreement will remain in full force and effect. The controlling language of this Agreement, and any proceedings relating to this Agreement, shall be English. You agree to bear any and all costs of translation, if necessary. The headings to the sections of this Agreement are used for convenience only and shall have no substantive meaning. All questions concerning this Agreement shall be directed to: Applied Biosystems, 850 Lincoln Centre Drive, Foster City, CA 94404-1128, Attention: Legal Department.

Applied Biosystems, LLC, 850 Lincoln Centre Drive, Foster City, CA 94404.

Documentation

Related documentation

Document	Part number	Description
Applied Biosystems SOLiD™ 4 System Library Preparation Guide	4445673	Describes how to prepare libraries.
Applied Biosystems SOLiD™ 4 System Library Preparation Quick Reference Card	4445674	Provides brief, step-by-step procedures for preparing libraries.
Applied Biosystems SOLiD™ 4 System Templated Bead Preparation Guide	4448378	Describes how to prepare templated beads by emulsion PCR (ePCR), required before sequencing on the SOLiD™ 4 System.
Applied Biosystems SOLiD™ 4 System Templated Bead Preparation Quick Reference Card	4448329	Provides brief, step-by-step procedures for preparing templated beads by emulsion PCR (ePCR), required before sequencing on the SOLiD™ 4 System.
Applied Biosystems SOLiD™ 4 System Instrument Operation Guide	4448379	Describes how to load and run the SOLiD™ 4 System for sequencing.
Applied Biosystems SOLiD™ 4 System Instrument Operation Quick Reference Card	4448380	Provides brief, step-by-step procedures for loading and running the SOLiD™ 4 System.
Applied Biosystems SOLiD™ 4 System Site Preparation Guide	4448639	Provides all the information that you need to set up the SOLiD™ 4 System.
Applied Biosystems SOLiD™ 4 System SETS Software User Guide	4448411	Provides an alternate platform to monitor runs, modify settings and reanalyze previous runs that are performed on the SOLiD System.
Applied Biosystems SOLiD™ 4 System ICS Software Help	—	Describes the software and provides procedures for common tasks (see the Instrument Control Software).

Document	Part number	Description
BioScope™ Software for Scientists Guide	4448431	Provides a bioinformatics analysis framework for flexible application analysis (data-generated mapping, SNPs, count reads) from sequencing runs.
Working with SOLiDBioScope.com™ Quick Reference Card	4452359	Provides an online suite of software tools for Next Generation Sequencing (NGS) analysis. SOLiDBioScope.com™ leverages the scalable resources of cloud computing to perform compute-intensive NGS data processing.
Applied Biosystems SOLiD™ 4 System Software Integrated Workflow Quick Reference Guide	4448432	Describes the relationship between the softwares comprising the SOLiD 4 platform and provides quick step procedures on operating each software to perform data analysis.
Applied Biosystems SOLiD™ 4 System Product Selection Guide	4452360	Provides a quick guide to the sequencing kits you need to perform fragment, paired-end, mate-pair, multiplex fragment, and multiplex paired-end sequencing.

Send us your comments

Applied Biosystems welcomes your comments and suggestions for improving its user documents. You can e-mail your comments to:

techpubs@appliedbiosystems.com

IMPORTANT! The e-mail address above is for submitting comments and suggestions relating *only* to documentation. To order documents, download PDF files, or for help with a technical question, see www.lifetechnologies.com.

Index

applied biosystems™
part of life technologies™

Applied Biosystems
850 Lincoln Centre Drive | Foster City, CA 94404 USA
Phone 650.638.5800 | Toll Free 800.345.5224
www.appliedbiosystems.com

Technical Resources and Support
For the latest technical resources and support information for all locations, please refer to our Web site at
www.appliedbiosystems.com/support